
UUU UUU EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT PPPPPPPPPPPP SSSSSSSSSSS YYY YYY
UUU UUU EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT PPPPPPPPPPPP SSSSSSSSSSS YYY YYY
UUU UUU EEEEEEEEEEEEEEE TTTTTTTTTTTTTTT PPPPPPPPPPPP SSSSSSSSSSS YYY YYY
UUU UUU EEE TTT PPP PPP SSS YYY YYY
UUU UUU EEE TTT PPP PPP SSS YYY YYY
UUU UUU EEE TTT PPP PPP SSS YYY YYY
UUU UUU EEE TTT PPP PPP SSS YYY YYY
UUU UUU EEE TTT PPP PPP SSS YYY YYY
UUU UUU EEE TTT PPP PPP SSS YYY YYY
UUU UUU EEEEEEEEEEE TTT PPPPPPPPPPPP SSSSSSSSS YYY
UUU UUU EEEEEEEEEEE TTT PPPPPPPPPPPP SSSSSSSSS YYY
UUU UUU EEEEEEEEEEE TTT PPPPPPPPPPPP SSSSSSSSS YYY
UUU UUU EEE TTT PPP SSS YYY
UUU UUU EEE TTT PPP SSS YYY
UUU UUU EEE TTT PPP SSS YYY
UUU UUU EEE TTT PPP SSS YYY
UUU UUU EEE TTT PPP SSS YYY
UUU UUU EEE TTT PPP SSS YYY
UUUUUUUUUUUUUUU EEEEEEEEEEEEEEE TTT PPP SSSSSSSSSSS YYY
UUUUUUUUUUUUUUU EEEEEEEEEEEEEEE TTT PPP SSSSSSSSSSS YYY
UUUUUUUUUUUUUUU EEEEEEEEEEEEEEE TTT PPP SSSSSSSSSSS YYY
UUUUUUUUUUUUUUU EEEEEEEEEEEEEEE TTT PPP SSSSSSSSSSS YYY

FILEID**UETINIT00

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT    IIIIII    NN      NN    IIIIII    TTTTTTTTTT    000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT    IIIIII    NN      NN    IIIIII    TTTTTTTTTT    000000      000000
UU      UU  EE              TT          II      NN      NN      II        TT        00      00  00      00
UU      UU  EE              TT          II      NN      NN      II        TT        00      00  00      00
UU      UU  EE              TT          II      NNNN    NN      II        TT        00    0000  00    0000
UU      UU  EE              TT          II      NNNN    NN      II        TT        00    0000  00    0000
UU      UU  EEEEEEEE        TT          II      NN  NN  NN      II        TT        00  00  00  00  00  00
UU      UU  EEEEEEEE        TT          II      NN  NN  NN      II        TT        00  00  00  00  00  00
UU      UU  EE              TT          II      NN    NNNN      II        TT        0000    00  0000    00
UU      UU  EE              TT          II      NN    NNNN      II        TT        0000    00  0000    00
UU      UU  EE              TT          II      NN      NN      II        TT        00      00  00      00
UU      UU  EE              TT          II      NN      NN      II        TT        00      00  00      00
UUUUUUUUUU  EEEEEEEEEE      TT        IIIIII    NN      NN    IIIIII      TT          000000      000000
UUUUUUUUUU  EEEEEEEEEE      TT        IIIIII    NN      NN    IIIIII      TT          000000      000000

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSS
```

Table of contents

UET
V04

6E
20
20
4C

6F
65
65
6C
73
43

29

```
0000     1          .TITLE UETINIT00 VAX/VMS UETP USER INTERFACE PROGRAM
0000     2          .IDENT  'V04-001'
0000     3          .ENABLE SUPPRESSION
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; FACILITY:
0000    31 ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    32 ;       account.
0000    33 ;
0000    34 ; ABSTRACT:
0000    35 ;       This program handles all UETP user interface dialogue.
0000    36 ;
0000    37 ; ENVIRONMENT:
0000    38 ;       This program requires the following privileges and quotas:
0000    39 ;               GRPNAM, CMEXEC
0000    40 ;
0000    41 ;--
0000    42 ;
0000    43 ; AUTHOR: Larry D. Jones,       CREATION DATE: November, 1980
0000    44 ;
0000    45 ; MODIFIED BY:
0000    46 ;
0000    47 ;       V04-001 RNH0015         Richard N. Holstein,    07-Sep-1984
0000    48 ;               Change BIOLM and ENQLM quotas to reflect new minima.
0000    49 ;
0000    50 ;       V03-015 RNH0014         Richard N. Holstein,    17-Aug-1984
0000    51 ;               Remove BYTLM quota check; BYTLM is used for WCBs.
0000    52 ;
0000    53 ;       V03-014 PEL0001         Patti E. Lutsky,        21-Jun-1984
0000    54 ;               Change reference to VENUS from 11/790 to 8600.
0000    55 ;
0000    56 ;       V03-013 RNH0013         Richard N. Holstein,    06-Mar-1984
0000    57 ;               Fix minor bugs in V03-011.
```

```
0000     58 ;
0000     59 ;       V03-012 KPL0100         Peter Lieberwirth       6-Mar-1984
0000     60 ;               Change CONFREG reference to CONFREGL.
0000     61 ;
0000     62 ;       V03-011 RNH0012         Richard N. Holstein,    27-Feb-1984
0000     63 ;               Take advantage of new UETP message codes.  Fix SSERROR
0000     64 ;               interaction with RMS_ERROR.  Get rid of SHOW MEMORY
0000     65 ;               subprocess in favor of new $GETSYI capabilities.  Incorporate
0000     66 ;               fixes from the device test template.  Rework message indicating
0000     67 ;               load test calculations.
0000     68 ;
0000     69 ;       V03-010 RNH0011         Richard N. Holstein,    02-Feb-1984
0000     70 ;               Allow a user to select any subset of UETP phases.  Remove the
0000     71 ;               "LOCAL" subset of phases as an option.  Remove old code which
0000     72 ;               was conditionally assembled in case we needed to include
0000     73 ;               non-paged pool in estimating loads.
0000     74 ;
0000     75 ;       V03-009 RNH0010         Richard N. Holstein,    01-Aug-1983
0000     76 ;               Fix bug in RNH0009 which picked the wrong CPU for variations
0000     77 ;               on a basic CPU type.
0000     78 ;
0000     79 ;       V03-008 RNH0009         Richard N. Holstein,    29-Jul-1983
0000     80 ;               Add CLUSTER and LOCAL "phase names".  Support new CPU types,
0000     81 ;               SUPERSTAR, VENUS, SCORPIO, NAUTILUS, SEAHORSE I, microVAX chip.
0000     82 ;
0000     83 ;       V03-007 RNH0008         Richard N. Holstein,    26-May-1983
0000     84 ;               Change ASTLM and DIOLM to 55, each.
0000     85 ;
0000     86 ;       V03-006 BAA0002         Brian A. Axtell,        14-Dec-1982
0000     87 ;               Removed phase names for RMS32, system services, native
0000     88 ;               utilities, and compatibility mode tests from phase inquire.
0000     89 ;
0000     90 ;       V03-005 BAA0001         Brian A. Axtell,        14-Dec-1982
0000     91 ;               Fixed problem when prompting for phase names so that it
0000     92 ;               doesn't drop a phase if there is an input error.
0000     93 ;
0000     94 ;       V03-004 RNH0007         Richard N. Holstein,    18-Oct-1982
0000     95 ;               Check for errors upon termination of the subprocess which
0000     96 ;               does a SHOW MEMORY command into a file.
0000     97 ;
0000     98 ;       V03-003 RNH0006         Richard N. Holstein,    12-Jul-1982
0000     99 ;               Change our dependency on SHOW MEMORY so that we expect a
0000    100 ;               second line of paging file info for shorter filespecs.
0000    101 ;
0000    102 ;       V03-002 LDJ0006         Larry D. Jones,         30-Mar-1982
0000    103 ;               Fix dump mode equation output, modified by history and
0000    104 ;               set the 11/782 cpu scale value.
0000    105 ;
0000    106 ;       V03-001 RNH0005         Richard N. Holstein,    23-Mar-1982
0000    107 ;               Fix confusing error message.
0000    108 ;
0000    109 ;**
```

```
0000   111          .SBTTL  Declarations
0000   112          .ENABLE SUPPRESSION
0000   113 ;
0000   114 ; INCLUDE FILES:
0000   115 ;
0000   116 ;          SYS$SYSTEM:SYS.STB                  ; To get EXE$GB_CPUTYPE
0000   117 ;          SYS$LIBRARY:LIB.MLB                 ; To get definitions
0000   118 ;          SHRLIB$:UETP.MLB                    ; To get UETP definitions
0000   119 ;
0000   120 ; MACROS:
0000   121 ;
0000   122          $ACCDEF                             ; Accounting info - termination mailbox
0000   123          $CHFDEF                             ; Condition handler frame definitions
0000   124          $CLIDEF                             ; CLI definitions
0000   125          $CLISERVDEF                         ; CLI callback definitions
0000   126          $JPIDEF                             ; $GETJPI definitions
0000   127          $NDTDEF                             ; SBI nexus definitions
0000   128          $PRDEF                              ; Processor register definitions
0000   129          $RPBDEF                             ; Restart parameter block definitions
0000   130          $SHRDEF                             ; Shared messages
0000   131          $STSDEF                             ; Status return
0000   132          $SYIDEF                             ; $GETSYI definitions
0000   133          $UETPDEF                            ; UETP
0000   134
0000   135          .MACRO ITMENT NAME,POSITION,EXPECTED
0000   136          .=PC1...
0000   137          .BYTE   ^X'POSITION                 ; Bit of priv or quota to check
0000   138          PC1...=PC1...+1
0000   139          .=PC2...
0000   140          .LONG   EXPECTED                    ; Expected results
0000   141          PC2...=PC2...+4
0000   142          .=PC3...
0000   143          .ADDRESS PC5...                     ; Address of priv or quota ASCIC name
0000   144          PC3...=PC3...+4
0000   145          .=PC5...
0000   146 NAME:                                       ; Ascic name
0000   147          .ASCIC  /NAME/
0000   148          PC5...=.
0000   149          .ENDM ITMENT
```

```
                     0000   151 ;
                     0000   152 ; EQUATED SYMBOLS:
                     0000   153
                     0000   154 ;   Facility number definitions:
00000001             0000   155         RMS$_FACILITY = 1
                     0000   156
                     0000   157 ;   SHR message definitions:
00740000             0000   158         UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
                     0000   159 ;
007410E0             0000   160         UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038             0000   161         UETP$_BEGIND = UETP!SHR$_BEGIND
00741080             0000   162         UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741130             0000   163         UETP$_TEXT   = UETP!SHR$_TEXT
00741108             0000   164         UETP$_BADKEY = UETP!SHR$_BADKEY
                     0000   165
                     0000   166 ;   Miscellany:
000000FF             0000   167         LOGNAM_SIZE    = 255                ; Maximum logical name size
00000004             0000   168         SYMBOL_CNT     = 4                  ; Number of local syms to be evaluated
0000012C             0000   169         TEXT_BUFFER    = 300                ; Internal text buffer size
000000FF             0000   170         MAXSYM_SZ      = 255                ; Maximum symbol size
0000000D             0000   171         CR             = ^XD                ; Carriage return
0000000A             0000   172         LF             = ^XA                ; Line feed
0000004D             0000   173         M              = ^A/M/              ; M character
00000020             0000   174         SPACE          = ^A/ /              ; Space character
00000009             0000   175         TAB            = ^A/     /          ; Tab character
00000020             0000   176         LCBIT          = ^X20               ; Lower case bit
00000001             0000   177         PROMPTV        = 1                  ; Flag set if must prompt for input
00000002             0000   178         PROMPTM        = 1@PROMPTV
00000002             0000   179         TERMINALV      = 2                  ; Flag set if SYS$COMMAND is a terminal
00000004             0000   180         TERMINALM      = 1@TERMINALV
00000003             0000   181         PRIV_PRNTV     = 3                  ; Flag set if already printed priv msg
00000004             0000   182         DUMPV          = 4                  ; Flag set if running in dump mode
00000010             0000   183         DUMPM          = 1@DUMPV
0000001E             0000   184         PRIV_CNT       = 30                 ; Privilege count
00000009             0000   185         QUOT_CNT       = 9                  ; Quota count
000003E8             0000   186         PP_PAGE_USAGE  = 1000               ; Est. of per process use of page & pool
CCCD3F4C             0000   187         PER_WS_INUSE = ^F0.20               ; Est. %age of proc continuous use of its WS
```

```
                                0000   189          .SBTTL  Read-Only Data
                            00000000   190          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                0000   191
                                0000   192 ACNT_NAME:                                ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'   0000   193          .ASCID  /SYSTEST/
                                  54   000E
                                000F   194
                                000F   195 TEST_NAME:                                ; This test name
49 4E 49 54 45 55 00000017'010E0000'   000F   196          .ASCID  /UETINITOO/
                              30 30 54 001D
                                0020   197
                                0020   198 MODE:                                     ; Run mode logical name
      45 44 4F 4D 00000028'010E0000'   0020   199          .ASCID  /MODE/
                                002C   200
                                002C   201 DUMP:                                     ; String to match...
      50 4D 55 44 00000034'010E0000'   002C   202          .ASCID  /DUMP/            ; ...if we're to run in dump mode
                                0038   203
                                0038   204 SYS$COMMAND:                              ; Name of device from which...
4F 43 24 53 59 53 00000040'010E0000'   0038   205          .ASCID  /SYS$COMMAND/     ; ...the test can be aborted
                        44 4E 41 4D 4D 0046
                                004B   206
                                004B   207 COMMAND_ITMLST:                          ; $GETDVI arg list for SYS$COMMAND
                         0000'0004     004B   208          .WORD   4,DVI$_DEVCLASS   ; We need the device class...
                    00000000 00000401' 004F   209          .LONG   DEVBUF,0
                         0000'0040     0057   210          .WORD   64,DVI$_DEVNAM    ; ...and the equivalence name
                    00000045'0000004D' 005B   211          .LONG   BUFFER,BUFFER_PTR
                          00000000     0063   212          .LONG   0                 ; Terminate the list
                                0067   213
                                0067   214 USERS:                                    ; Load count logical name
   53 44 41 4F 4C 0000006F'010E0000'   0067   215          .ASCID  /LOADS/
                                0074   216
                                0074   217 PASS_NAME:                               ; Local pass count logical name
4E 43 53 53 41 50 0000007C'010E0000'   0074   218          .ASCID  /PASSCNT/
                                  54   0082
                                0083   219
                                0083   220 REPORT_NAME:                             ; Long or short report indicator name
54 52 4F 50 45 52 0000008B'010E0000'   0083   221          .ASCID  /REPORT/
                                0091   222
                                0091   223 SYSDISK:                                 ; Name of device we are booted from
59 53 24 53 59 53 00000099'010E0000'   0091   224          .ASCID  /SYS$SYSROOT/
                        54 4F 4F 52 53 009F
                                00A4   225
                                00A4   226 NO_RMS_AST_TABLE:                        ; List of errors for which...
                          00000000'    00A4   227          .LONG   RMS$_BLN          ; ...RMS cannot deliver an AST...
                          00000000'    00A8   228          .LONG   RMS$_BUSY         ; ...even if one has an ERR= arg
                          00000000'    00AC   229          .LONG   RMS$_CDA          ; Note that we can search table...
                          00000000'    00B0   230          .LONG   RMS$_FAB          ; ...via MATCHC since <31:16>...
                          00000000'    00B4   231          .LONG   RMS$_RAB          ; ...pattern can't be in <15:0>
                          00000014     00B8   232 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                00B8   233
                                00B8   234 CNTRLCMSG:
65 74 72 6F 62 41 000000C0'010E0000'   00B8   235          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64    00C6
                  43 2F 4C 52 54 43 20 00D2
                                00D9   236
                                00D9   237 SYNTAX_ERROR_MSG:
78 61 74 6E 79 53 000000E1'010E0000'   00D9   238          .ASCID  /Syntax error in response.  Please try again./
```

```
65 72 20 6E 69 20 72 6F 72 72 65 20  00E7
65 6C 50 20 20 2E 65 73 6E 6F 70 73  00F3
69 61 67 61 20 79 72 74 20 65 73 61  00FF
                        2E 6E       010B
                                    010D   239
                                    010D   240 INVALID_PHASE_MSG:
73 69 20 53 41 21 00000115'010E0000'  010D   241         .ASCID  /!AS is not a valid phase name!/
64 69 6C 61 76 20 61 20 74 6F 6E 20  011B
21 65 6D 61 6E 20 65 73 61 68 70 20  0127
                                    0133   242
                                    0133   243 INVALID_PASS_MSG:
73 69 20 53 41 21 0000013B'010E0000'  0133   244         .ASCID  /!AS is not a valid pass count!/
64 69 6C 61 76 20 61 20 74 6F 6E 20  0141
21 74 6E 75 6F 63 20 73 73 61 70 20  014D
                                    0159   245
                                    0159   246 INVALID_LOADCNT_MSG:
73 69 20 53 41 21 00000161'010E0000'  0159   247         .ASCID  /!AS is not a valid load count!/
64 69 6C 61 76 20 61 20 74 6F 6E 20  0167
21 74 6E 75 6F 63 20 64 61 6F 6C 20  0173
                                    017F   248
                                    017F   249 INVALID_REPORT_MSG:
73 69 20 53 41 21 00000187'010E0000'  017F   250         .ASCID  /!AS is not a valid report type!/
64 69 6C 61 76 20 61 20 74 6F 6E 20  018D
65 70 79 74 20 74 72 6F 70 65 72 20  0199
                                 21 01A5
                                    01A6   251
                                    01A6   252 COMMAND_DVI_FAILED:
56 44 54 45 47 24 000001AE'010E0000'  01A6   253         .ASCID  \$GETDVI failed for SYS$COMMAND.  Status returned was:\
72 6F 66 20 64 65 6C 69 61 66 20 49  01B4
44 4E 41 4D 4D 4F 43 24 53 59 53 20  01C0
65 72 20 73 75 74 61 74 53 20 20 2E  01CC
   3A 73 61 77 20 64 65 6E 72 75 74  01D8
                                    01E3   254
                                    01E3   255 WRONG_ACCOUNT:
72 61 20 75 6F 59 000001EB'010E0000'  01E3   256         .ASCID  \You are logged into the wrong account.\<CR><LF>-
74 6E 69 20 64 65 67 67 6F 6C 20 65  01F1
20 67 6E 6F 72 77 20 65 68 74 20 6F  01FD
         0D 2E 74 6E 75 6F 63 63 61  0209
67 6F 6C 20 65 73 61 65 6C 50 09 0A  0212   257                 \       Please login to the SYSTEST account.\
59 53 20 65 68 74 20 6F 74 20 6E 69  021E
6E 75 6F 63 63 61 20 54 53 45 54 53  022A
                              2E 74  0236
                                    0238   258
                                    0238   259 STRSTR:
6F 66 20 65 68 54 00000240'010E0000'  0238   260         .ASCID  \The following:\<CR><LF>
      0A 0D 3A 67 6E 69 77 6F 6C 6C  0246
                                    0250   261
                                    0250   262 ENDSTR:
20 65 72 61 0A 0D 00000258'010E0000'  0250   263         .ASCID  <CR><LF>\are non-standard for the SYSTEST account and may\-
64 72 61 64 6E 61 74 73 2D 6E 6F 6E  025E
53 59 53 20 65 68 74 20 72 6F 66 20  026A
74 6E 75 6F 63 63 61 20 54 53 45 54  0276
            79 61 6D 20 64 6E 61 20  0282
55 20 6E 69 20 74 6C 75 73 65 72 20  028A   264                 \ result in UETP errors.\
   2E 73 72 6F 72 72 65 20 50 54 45  0296
                                    02A1   265
                                    02A1   266 CTRSTR:
```

```
20 43 41 21 5F 21 000002A9'010E0000'  02A1   267          .ASCID  \!_!AC !AC,\
                        2C 43 41 21   02AF
                                      02B3   268 PRV_STR:
      65 67 65 6C 69 76 69 72 70 00'  02B3   269          .ASCIC  \privilege\
                                 09   02B3
                                      02BD   270 QUO_STR:
            61 74 6F 75 71 00'        02BD   271          .ASCIC  \quota\
                                 05   02BD
                                      02C3   272 FILE:                                ; Fills in RMS_ERR_STRING
      65 6C 69 66 000002CB'010E0000'  02C3   273          .ASCID  /file/
                                      02CF   274
                                      02CF   275 RECORD:                              ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 000002D7'010E0000'  02CF   276          .ASCID  /record/
                                      02DD   277
                                      02DD   278 RMS_ERR_STRING:                      ; Announces an RMS error
41 21 20 53 4D 52 000002E5'010E0000'  02DD   279          .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53   02EB
            44 41 21 20 65 6C 69      02F7
                                      02FE   280 SYSTEM:
20 75 6F 59 2F 21 00000306'010E0000'  02FE   281          .ASCID  \!/You are running on an !AC CPU with !UL pages of memory.\
20 67 6E 69 6E 6E 75 72 20 65 72 61   030C
50 43 20 43 41 21 20 6E 61 20 6E 6F   0318
70 20 4C 55 21 20 68 74 69 77 20 55   0324
6F 6D 65 6D 20 66 6F 20 73 65 67 61   0330
                        2E 79 72      033C
                                      033F   282 DISK:
79 73 20 65 68 54 00000347'010E0000'  033F   283          .ASCID  \The system was booted from !AS.\
6F 6F 62 20 73 61 77 20 6D 65 74 73   034D
53 41 21 20 6D 6F 72 66 20 64 65 74   0359
                              2E      0365
                                      0366   284 PASS_PROMPT:
61 6D 20 77 6F 48 0000036E'010E0000'  0366   285          .ASCID  \How many passes of UETP do you wish to run [1]? \
66 6F 20 73 65 73 73 61 70 20 79 6E   0374
75 6F 79 20 6F 64 20 50 54 45 55 20   0380
6E 75 72 20 6F 74 20 68 73 69 77 20   038C
                  20 3F 5D 31 5B 20   0398
                                      039E   286 LOAD_PROMPT:
61 6D 20 77 6F 48 000003A6'010E0000'  039E   287          .ASCID  \How many simulated user loads do you want [!UL]? \
64 65 74 61 6C 75 6D 69 73 20 79 6E   03AC
20 73 64 61 6F 6C 20 72 65 73 75 20   03B8
20 74 6E 61 77 20 75 6F 79 20 6F 64   03C4
                  20 3F 5D 4C 55 21 5B 03D0
                                      03D7   288 REPORT_PROMPT:
75 6F 79 20 6F 44 000003DF'010E0000'  03D7   289          .ASCID  \Do you want Long or Short report format [Long]? \
6F 20 67 6E 6F 4C 20 74 6E 61 77 20   03E5
6F 70 65 72 20 74 72 6F 68 53 20 72   03F1
4C 5B 20 74 61 6D 72 6F 66 20 74 72   03FD
                  20 3F 5D 67 6E 6F   0409
                                      040F   290 START_MESSAGE:
50 54 45 55 2F 21 00000417'010E0000'  040F   291          .ASCID  \!/UETP starting at !%D with parameters:\
74 61 20 67 6E 69 74 72 61 74 73 20   041D
61 70 20 68 74 69 77 20 44 25 21 20   0429
            3A 73 72 65 74 65 6D 61 72 0435
                                      043E   292 PHASES:
65 73 61 68 70 20 00000446'010E0000'  043E   293          .ASCID  \ phases\
                                    73 044C
                                      044D   294 LONG_MSG:
```

```
                                                                                295          .ASCID  /, long report./<CR><LF>
                                                                                296 SHORT_MSG:
                                                                                297          .ASCID  /, short report./<CR><LF>
                                                                                298
                                                                                299 DUMP_MSG_PTR:                                ; $PUTMSG MSGVEC for load calc msgs
                                                0001 0003  047E                 300          .WORD   3,1
                                                00741131   0482                 301          .LONG   UETP$_TEXT!STS$K_SUCCESS
                                                00000001   0486                 302          .LONG   1
                                                00000045'  048A                 303          .ADDRESS BUFFER_PTR
                                                           048E                 304
                                                           048E                 305 DUMP_MSG1:
20 65 68 54 2F 21 00000496'010E0000'  048E      306          .ASCID  \!/The default number of loads is the minimum result of!/!/\-
62 6D 75 6E 20 74 6C 75 61 66 65 64   049C
20 73 64 61 6F 6C 20 66 6F 20 72 65   04A8
6D 69 6E 69 6D 20 65 68 74 20 73 69   04B4
66 6F 20 74 6C 75 73 65 72 20 6D 75   04C0
                  2F 21 2F 21         04CC
45 4C 41 43 53 5F 55 50 43 20 29 31   04D0      307          \1) CPU_SCALE * ((MEM_FREE + MEM_MODIFY) / (WS_SIZE * PER_WS_INUSE))!/\-
45 52 46 5F 4D 45 4D 28 28 20 2A 20   04DC
49 44 4F 4D 5F 4D 45 4D 20 2B 20 45   04E8
49 53 5F 53 57 28 20 2F 20 29 59 46   04F4
5F 53 57 5F 52 45 50 20 2A 20 45 5A   0500
         2F 21 29 29 45 53 55 4E 49   050C
20 53 41 21 20 20 20 20 20 20 20 20   0515      308          \        !AS * ((!8UL + !10UL) / (!7UL *          !4AS))  = !UL!/\
21 20 2B 20 4C 55 58 21 28 28 20 2A   0521
55 37 21 28 20 2F 20 29 4C 55 50 31   052D
20 20 20 20 20 20 20 20 20 2A 20 4C   0539
55 21 20 3D 20 20 29 29 53 41 54 21   0545
                        2F 21 4C      0551
                                      0554 309 DUMP_MSG2:
65 72 46 20 29 32 0000055C'010E0000'  0554      310          .ASCID  \2) Free process slots                                  \-
6C 73 20 73 73 65 63 6F 72 70 20 65   0562
20 20 20 20 20 20 20 20 20 73 74 6F   056E
20 20 20 20 20 20 20 20 20 20 20 20   057A
20 20 20 20 20 20 20 20 20 20 20 20   0586
                  20 20 20 20 20 20   0592
21 20 3D 20 20 20 20 20 20 20 20 20   0598      311          \          = !UL!/!/\-
                  2F 21 2F 21 4C 55   05A4
65 67 61 70 20 65 65 72 46 20 29 33   05AA      312          \3) Free page file pages / Typical use of page file pages per process!/\-
20 73 65 67 61 70 20 65 6C 69 66 20   05B6
73 75 20 6C 61 63 69 70 79 54 20 2F   05C2
69 66 20 65 67 61 70 20 66 6F 20 65   05CE
72 65 70 20 73 65 67 61 70 20 65 6C   05DA
      2F 21 73 73 65 63 6F 72 70 20   05E6
55 32 34 21 20 2F 20 4C 55 33 32 21   05F0      313          \!23UL / !42UL = !UL!/\
         2F 21 4C 55 21 20 3D 20 4C   05FC
                                      0605 314
                                      0605 315 LOGINOUT:                               ; Name of login image
59 53 24 53 59 53 0000060D'010E0000'  0605      316          .ASCID  /SYS$SYSTEM:LOGINOUT.EXE/
55 4F 4E 49 47 4F 4C 3A 4D 45 54 53   0613
                     45 58 45 2E 54   061F
                                      0624 317
                                      0624 318 OFFSET:                                 ; Offset table
                        0000064B      0624 319          .BLKB   PRIV_CNT+QUOT_CNT
                                      064B 320 EXPECTED:                               ; Results expected table
```

```
                    000006E7  0648   321          .BLKL   PRIV_CNT+QUOT_CNT
                              06E7   322  NAM_PTRS:                        ; Name pointer table
                    00000783  06E7   323          .BLKL   PRIV_CNT+QUOT_CNT
                              0783   324  NAME_TBL:                       ; ASCIC name table
                    00000624  0783   325          PC1... = OFFSET
                    0000064B  0783   326          PC2... = EXPECTED
                    000006E7  0783   327          PC3... = NAM_PTRS
                    00000783  0783   328          PC5... = .
                              0783   329          .LIST MEB
                              0783   330          ITMENT ALLSPOOL , 04, 0        ; Privilege entries
                    00000624  0783                .=PC1...
                          04  0624                .BYTE   ^X04             ; Bit of priv or quota to check
                    0000064B  0625                .=PC2...
                    00000000  064B                .LONG   0                ; 0 results
                    000006E7  064F                .=PC3...
                    00000783' 06E7                .ADDRESS PC5...                ; Address of priv or quota ASCIC ALLSPOOL
                    00000783  06EB                .=PC5...
                              0783          ALLSPOOL:                              ; Ascic ALLSPOOL
 4C 4F 4F 50 53 4C 4C 41 00'  0783                .ASCIC  /ALLSPOOL/
                           08  0783
                              078C   331          .NLIST MEB
                              078C   332          ITMENT BUGCHK    , 17, 0
                              0793   333          ITMENT BYPASS    , 1D, 0
                              079A   334          ITMENT CMEXEC    , 01, 1
                              07A1   335          ITMENT CMKRNL    , 00, 1
                              07A8   336          ITMENT DETACH    , 05, 1
                              07AF   337          ITMENT DIAGNOSE  , 06, 1
                              07B8   338          ITMENT EXQUOTA   , 13, 0
                              07C0   339          ITMENT GROUP     , 08, 1
                              07C6   340          ITMENT GRPNAM    , 03, 1
                              07CD   341          ITMENT LOG_IO    , 07, 1
                              07D4   342          ITMENT MOUNT     , 11, 0
                              07DA   343          ITMENT NETMBX    , 14, 1
                              07E1   344          ITMENT NOACNT    , 09, 0
                              07E8   345          ITMENT OPER      , 12, 0
                              07ED   346          ITMENT PFNMAP    , 1A, 0
                              07F4   347          ITMENT PHY_IO    , 16, 1
                              07FB   348          ITMENT PRMCEB    , 0A, 1
                              0802   349          ITMENT PRMGBL    , 18, 0
                              0809   350          ITMENT PRMMBX    , 0B, 1
                              0810   351          ITMENT PSWAPM    , 0C, 0
                              0817   352          ITMENT SETPRI    , 0D, 0
                              081E   353          ITMENT SETPRV    , 0E, 1
                              0825   354          ITMENT SHMEM     , 1B, 0
                              082B   355          ITMENT SYSGBL    , 19, 0
                              0832   356          ITMENT SYSNAM    , 02, 1
                              0839   357          ITMENT SYSPRV    , 1C, 1
                              0840   358          ITMENT TMPMBX    , 0F, 1
                              0847   359          ITMENT VOLPRO    , 15, 1
                              084E   360          ITMENT WORLD     , 10, 0
                              0854   361
                              0854   362          ITMENT ASTLM     , 00, 55       ; Quota entries
                              085A   363          ITMENT BIOLM     , 01, 18
                              0860   364          ITMENT CPULIM    , 03, 0
                              0867   365          ITMENT ENQLM     , 04, 30
                              086D   366          ITMENT DIOLM     , 05, 55
                              0873   367          ITMENT FILLM     , 06, 20
```

```
                     0879    368          ITMENT PGFLQUOTA, 07, 10000
                     0883    369          ITMENT PRCLM    , 08, 8
                     0889    370          ITMENT TQLM     , 09, 20
                     088E    371
                     088E    372  GETSYI_ITMLST:                          ; $GETSYI arg list for...
           1001 0004 088E    373          .WORD   4,SYI$_SID              ; ...SID register...
00000000'000009F7' 0892    374          .ADDRESS SID,0
           10F4 0004 089A    375          .WORD   4,SYI$_PAGEFILE_FREE    ; ...space remaining in page file(s)
00000000'000009F3' 089E    376          .ADDRESS PAGE_SIZE,0
           00000000 08A6    377          .LONG   0
                     08AA    378
                     08AA    379  ; NOTE:  The code which searches CPU tables should limit itself to looking at
                     08AA    380  ; PR$_SID_TYPMAX (+1, to include illegal or unknown entries) entries.  In
                     08AA    381  ; order to prepare for planned CPUs though, we define a constant, CTT_LENGTH,
                     08AA    382  ; based on what we know is down the road.  This constant in the code must also
                     08AA    383  ; be patched if entries for new CPUs are patched in.
                     08AA    384  ;
                     08AA    385  ; Negative entries in the following tables apply to CPUs for which there is no
                     08AA    386  ; explicit CPU type defined, e.g., tightly coupled, multiple CPU configurations
                     08AA    387  ; such as the 11/782, or jacked up CPUs like the 11/785.
                     08AA    388
                     08AA    389  ; No negative entries for this table
                     08AA    390  CPU_TYPE_TABLE:                         ; Table of known CPU types
          00         08AA    391          .BYTE   0                       ; Illegal or unknown type
          01         08AB    392          .BYTE   PR$_SID_TYP780          ; STAR
          02         08AC    393          .BYTE   PR$_SID_TYP750          ; COMET
          03         08AD    394          .BYTE   PR$_SID_TYP730          ; NEBULA
          04         08AE    395          .BYTE   PR$_SID_TYP790          ; VENUS
          05         08AF    396          .BYTE   5                       ; SCORPIO (reserved)
          06         08B0    397          .BYTE   6                       ; NAUTILUS (reserved)
          07         08B1    398          .BYTE   PR$_SID_TYPUV1          ; SEAHORSE I
          08         08B2    399          .BYTE   PR$_SID_TYPUV2          ; microVAX chip
00000009             08B3    400  CTT_LENGTH = .-CPU_TYPE_TABLE           ; Item count of known CPUs + unknown
00000885             08B3    401          .BLKB   2                       ; Expansion room for new CPU's
                     08B5    402  ; End of CPU_TYPE_TABLE
                     08B5    403
                     08B5    404  ; Negative entries for CPU_NAME_TABLE
00000009             08B5    405          .BLKA   9                       ; Expansion for new CPU configurations
000009C7'            08D9    406          .ADDRESS A787                   ; Dual SUPERSTAR
000009C0'            08DD    407          .ADDRESS A785                   ; SUPERSTAR
00009B9'             08E1    408          .ADDRESS A782                   ; ATLAS
                     08E5    409  CPU_NAME_TABLE:                         ; CPU names address table
0000096D'            08E5    410          .ADDRESS UNKNOWN_CPU            ; Illegal or unknown CPU type
00000975'            08E9    411          .ADDRESS A780                   ; STAR
0000097C'            08ED    412          .ADDRESS A750                   ; COMET
00000983'            08F1    413          .ADDRESS A730                   ; NEBULA
0000098A'            08F5    414          .ADDRESS A8600                  ; VENUS
0000098F'            08F9    415          .ADDRESS ASCORPIO               ; SCORPIO
00000997'            08FD    416          .ADDRESS ANAUTILUS              ; NAUTILUS
000009A0'            0901    417          .ADDRESS AUV1                   ; SEAHORSE I
000009AB'            0905    418          .ADDRESS AUV2                   ; microVAX chip
00000911             0909    419          .BLKA   2                       ; Expansion room for new CPUs
                     0911    420  ; End of CPU_NAME_TABLE
                     0911    421
                     0911    422  ; Negative entries for CPU_SCALE_TABLE
00000935             0911    423          .BLKF   9                       ; Expansion for new CPU configurations
00004110             0935    424          .FLOAT  2.25                    ; Dual SUPERSTAR
```

```
                    000040C0  0939              .FLOAT  1.5                 ; SUPERSTAR
                    333340B3  093D              .FLOAT  1.4                 ; ATLAS
                              0941  CPU_SCALE_TABLE:                        ; Scale to balance loads vs CPU perf
                    00004080  0941              .FLOAT  1.0                 ; Illegal or unknown CPU
                    00004080  0945              .FLOAT  1.0                 ; STAR
                    CCCD404C  0949              .FLOAT  0.8                 ; COMET
                    00004000  094D              .FLOAT  0.5                 ; NEBULA
                    00004180  0951              .FLOAT  4.0                 ; VENUS
                    00004080  0955              .FLOAT  1.0                 ; SCORPIO
                    00004080  0959              .FLOAT  1.0                 ; NAUTILUS
                    00004080  095D              .FLOAT  1.0                 ; SEAHORSE I
                    00004080  0961              .FLOAT  1.0                 ; microVAX chip
                    0000096D  0965              .BLKF   2                   ; Expansion room for new CPUs
                              096D  ; End of CPU_SCALE_TABLE
                              096D
                              096D  UNKNOWN_CPU:                            ; Illegal or unknown CPU
        4E 57 4F 4E 4B 4E 55 00'  096D          .ASCIC  \UNKNOWN\
                          07  096D
                              0975  A780:                                   ; STAR
           30 38 37 2F 31 31 00'  0975          .ASCIC  \11/780\
                          06  0975
                              097C  A750:                                   ; COMET
           30 35 37 2F 31 31 00'  097C          .ASCIC  \11/750\
                          06  097C
                              0983  A730:                                   ; NEBULA
           30 33 37 2F 31 31 00'  0983          .ASCIC  \11/730\
                          06  0983
                              098A  A8600:                                  ; VENUS
                 30 30 36 38 00'  098A          .ASCIC  \8600\
                          04  098A
                              098F  ASCORPIO:                               ; SCORPIO
        4F 49 50 52 4F 43 53 00'  098F          .ASCIC  \SCORPIO\
                          07  098F
                              0997  ANAUTILUS:                              ; NAUTILUS
     53 55 4C 49 54 55 41 4E 00'  0997          .ASCIC  \NAUTILUS\
                          08  0997
                              09A0  AUV1:                                   ; SEAHORSE I
  49 20 45 53 52 4F 48 41 45 53 00'  09A0       .ASCIC  \SEAHORSE I\
                          0A  09A0
                              09AB  AUV2:                                   ; microVAX chip
68 63 20 58 41 56 6F 72 63 69 6D 00'  09AB      .ASCIC  \microVAX chip\
                       70 69  09B7
                          0D  09AB
                              09B9  A782:                                   ; ATLAS
           32 38 37 2F 31 31 00'  09B9          .ASCIC  \11/782\
                          06  09B9
                              09C0  A785:                                   ; SUPERSTAR
           35 38 37 2F 31 31 00'  09C0          .ASCIC  \11/785\
                          06  09C0
                              09C7  A787:                                   ; Dual SUPERSTAR
           37 38 37 2F 31 31 00'  09C7          .ASCIC  \11/787\
                          06  09C7
                              09CE
                              09CE  USER_LIST:                              ; GETJPI item list for USERNAME and WS size
                        000C  09CE              .WORD   12
                        0202  09D0              .WORD   JPI$_USERNAME
                   0000004D'  09D2              .LONG   BUFFER
```

```
00000988' 09D6   469          .LONG   OUTLEN
     0004 09DA   470          .WORD   4
     0402 09DC   471          .WORD   JPI$_WSQUOTA
000009AF' 09DE   472          .LONG   WS_SIZE
00000000  09E2   473          .LONG   0
     0004 09E6   474          .WORD   4
     0409 09E8   475          .WORD   JPI$_ASTLM
000009B3' 09EA   476          .LONG   JPI_ASTLM
00000000  09EE   477          .LONG   0
     0004 09F2   478          .WORD   4
     0310 09F4   479          .WORD   JPI$_BIOLM
000009B7' 09F6   480          .LONG   JPI_BIOLM
00000000  09FA   481          .LONG   0
     0004 09FE   482          .WORD   4
     031A 0A00   483          .WORD   JPI$_BYTLM
000009B8' 0A02   484          .LONG   JPI_BYTLM
00000000  0A06   485          .LONG   0
     0004 0A0A   486          .WORD   4
     040D 0A0C   487          .WORD   JPI$_CPULIM
000009BF' 0A0E   488          .LONG   JPI_CPULIM
00000000  0A12   489          .LONG   0
     0004 0A16   490          .WORD   4
     0320 0A18   491          .WORD   JPI$_ENQLM
000009C3' 0A1A   492          .LONG   JPI_ENQLM
00000000  0A1E   493          .LONG   0
     0004 0A22   494          .WORD   4
     0313 0A24   495          .WORD   JPI$_DIOLM
000009C7' 0A26   496          .LONG   JPI_DIOLM
00000000  0A2A   497          .LONG   0
     0004 0A2E   498          .WORD   4
     040F 0A30   499          .WORD   JPI$_FILLM
000009CB' 0A32   500          .LONG   JPI_FILLM
00000000  0A36   501          .LONG   0
     0004 0A3A   502          .WORD   4
     040E 0A3C   503          .WORD   JPI$_PGFLQUOTA
000009CF' 0A3E   504          .LONG   JPI_PGFLQUOTA
00000000  0A42   505          .LONG   0
     0004 0A46   506          .WORD   4
     0408 0A48   507          .WORD   JPI$_PRCLM
000009D3' 0A4A   508          .LONG   JPI_PRCLM
00000000  0A4E   509          .LONG   0
     0004 0A52   510          .WORD   4
     0410 0A54   511          .WORD   JPI$_TQLM
000009D7' 0A56   512          .LONG   JPI_TQLM
00000000  0A5A   513          .LONG   0
     0008 0A5E   514          .WORD   8
     0400 0A60   515          .WORD   JPI$_CURPRIV
000009DB' 0A62   516          .LONG   PRIV5
00000000  0A66   517          .LONG   0
00000000  0A6A   518          .LONG   0
          0A6E   519
          0A6E   520 SYM_NAM_TABLE:  ; Names of parameters in local symbol table
          0A6E   521                                        ; If defined they represent:
          0A6E   522 SYM_P1:                                ; phase
0000 0002' 0A6E   523          .WORD   P1_LEN,0
 00000A8E' 0A72   524          .ADDRESS P1_NAM
          0A76   525 SYM_P2:                                ; pass count
```

```
                     0000 0002'  0A76    526          .WORD   P2_LEN,0
                      00000A90'  0A7A    527          .ADDRESS P2_NAM
                                 0A7E    528  SYM_P3:                                 ; number of loads
                     0000 0002'  0A7E    529          .WORD   P3_LEN,0
                      00000A92'  0A82    530          .ADDRESS P3_NAM
                                 0A86    531  SYM_P4:                                 ; long or short report
                     0000 0002'  0A86    532          .WORD   P4_LEN,0
                      00000A94'  0A8A    533          .ADDRESS P4_NAM
                                 0A8E    534  P1_NAM:
                        31 50    0A8E    535          .ASCII  /P1/
                      00000002   0A90    536          P1_LEN = .-P1_NAM
                                 0A90    537  P2_NAM:
                        32 50    0A90    538          .ASCII  /P2/
                      00000002   0A92    539          P2_LEN = .-P2_NAM
                                 0A92    540  P3_NAM:
                        33 50    0A92    541          .ASCII  /P3/
                      00000002   0A94    542          P3_LEN = .-P3_NAM
                                 0A94    543  P4_NAM:
                        34 50    0A94    544          .ASCII  /P4/
                      00000002   0A96    545          P4_LEN = .-P4_NAM
                                 0A96    546
                                 0A96    547  PHASE_PROMPT:                           ; See if full UETP run is wanted
6E 75 52 0A 0A 0D 00000A9E'010E0000'  0A96    548          .ASCID  <CR><LF><LF>\Run "ALL" UETP phases or a "SUBSET" [ALL]? \
20 50 54 45 55 20 22 4C 4C 41 22 20   0AA4
20 61 20 72 6F 20 73 65 73 61 68 70   0AB0
4C 41 5B 20 22 54 45 53 42 55 53 22   0ABC
                  20 3F 5D 4C         0AC8
                                 0ACC    549
                                 0ACC    550  COMMA_BLANK:                           ; Separator between phase names...
                     20 2C 00'   0ACC    551          .ASCIC  \, \                    ; ...for WHICH_PHASE $FAOL string
                           02    0ACC
                                 0ACF    552
                                 0ACF    553  NEW_LINE:                              ; Continue list of phase names...
                  09 0A 0D 00'   0ACF    554          .ASCIC  <CR><LF>\       \       ; ...on a new line
                           03    0ACF
                                 0AD3    555
                                 0AD3    556  WHICH_PHASE1:                          ; Allow selection of UETP phases
                                 0AD3    557          .ASCID -
6F 59 2F 21 2F 21 00000ADB'010E0000'  0AD3    558          \!/!/You can choose one or more of the following phases:!/!/!_!#(AC)\
65 73 6F 6F 68 63 20 6E 61 63 20 75   0AE1
65 72 6F 6D 20 72 6F 20 65 6E 6F 20   0AED
6C 6C 6F 66 20 65 68 74 20 66 6F 20   0AF9
73 65 73 61 68 70 20 67 6E 69 77 6F   0B05
43 41 28 23 21 5F 21 2F 21 2F 21 3A   0B11
                              29      0B1D
                                 0B1E    559
                                 0B1E    560  WHICH_PHASE2:
29 73 28 65 73 61 68 50 0A 0A 0D 00'  0B1E    561          .ASCIC  <CR><LF><LF>\Phase(s):  \
                        20 20 3A      0B2A
                              0E      0B1E
                                 0B2D    562
                                 0B2D    563  ; We here take advantage of the Run Time Library $LIB_KEY_TABLE's internal
                                 0B2D    564  ; code so that we may generate descriptors of the keyword strings in
                                 0B2D    565  ; parallel with generating the strings and their pointers.  The sequence
                                 0B2D    566  ; of .ERROR statements below should guard us against internal changes to
                                 0B2D    567  ; the documented macro.
                                 0B2D    568
```

```
                     0B2D    569 ;LIB$$K_NPAIRS counts entries in $LIB_KEY_TABLE
                     0B2D    570 .MACRO          $$LIB_KEY_ENTRY STRING, VALUE
                     0B2D    571 .IF EQ LIB$$A_HERE+1    ; First time expanding this macro, define new stuff
                     0B2D    572         UETP$$A_THERE = LIB$$A_STRLOC
                     0B2D    573         LIB$$A_STRLOC = LIB$$A_STRLOC + <8 * LIB$$K_NPAIRS>
                     0B2D    574 .ENDC ; EQ LIB$$A_HERE+1
                     0B2D    575                 .ADDRESS        LIB$$A_STRLOC
                     0B2D    576                 .LONG           VALUE
                     0B2D    577         LIB$$A_HERE=.
                     0B2D    578         .=UETP$$A_THERE
                     0B2D    579         KEY_'STRING' DESC:
                     0B2D    580                 .WORD           %LENGTH(STRING),0
                     0B2D    581                 .ADDRESS        LIB$$A_STRLOC + 1       ; 1 char into ASCIC str
                     0B2D    582         UETP$$A_THERE=.
                     0B2D    583         .=LIB$$A_STRLOC
                     0B2D    584                 .ASCIC          \STRING\
                     0B2D    585         LIB$$A_STRLOC=.
                     0B2D    586         .=LIB$$A_HERE
                     0B2D    587 .ENDM           $$LIB_KEY_ENTRY
                     0B2D    588
            FFFFFFFF 0B2D    589 LIB$$A_HERE=-1                          ; Flags first $$LIB_KEY_ENTRY expansion
                     0B2D    590 SELECT_PHASE:                           ; Allow user to select between ALL...
                     0B2D    591         $LIB_KEY_TABLE  < -             ; ...phases or a subset of them
                     0B2D    592                 <ALL,0> -
                     0B2D    593                 <SUBSET,1> -
                     0B2D    594         > ; End of $LIB_KEY_TABLE
                     0B5C    595
            FFFFFFFF 0B5C    596 LIB$$A_HERE=-1                          ; Flags first $$LIB_KEY_ENTRY expansion
                     0B5C    597 PHASE_TABLE:
                     0B5C    598         $LIB_KEY_TABLE  < -
                     0B5C    599                 <<DEVICE>,>-
                     0B5C    600                 <<LOAD>,>-
                     0B5C    601                 <<DECNET>,>-
                     0B5C    602                 <<CLUSTER>,>-
                     0B5C    603         > ; End of $LIB_KEY_TABLE
                     0B88    604
                00000001 0B88    605 .MDELETE         $$LIB_KEY_ENTRY        ; Remove our own version of the macro
                     0B88    606 .IF NDF UETP$$A_THERE
                     0B88    607         .ERROR  ; This program depends on the existence
                     0B88    608         .ERROR  ; of $$LIB_KEY_ENTRY within the $LIB_KEY_TABLE
                     0B88    609         .ERROR  ; definition.  It must be fixed to use some
                     0B88    610         .ERROR  ; new definition so that it can generate
                     0B88    611         .ERROR  ; tables parallel to the ones from
                     0B88    612         .ERROR  ; $LIB_KEY_TABLE.
                     0B88    613 .ENDC ; NDF UETP$$A_THERE
                     0B88    614
                     0B88    615 UETPPHASE:                              ; Logical name for UETP.COM phase names
48 50 50 54 45 55 00000BC3'010E0000'  0B88    616         .ASCID  \UETPPHASE\
                  45 53 41  0BC9
```

```
                                0BCC   618          .SBTTL  Read/Write Data
                          00000000   619          .PSECT  RWDATA,WRT,NOEXE,PAGE
                                0000   620
                                0000   621 WELCOME:
                    0000002F'   0000   622          .LONG   WELCOML
                    00000008'   0004   623          .ADDRESS .+4
65 6D 6F 63 6C 65 57 09 0A  0A 0A 0D    0008   624          .ASCII  <CR><LF><LF><LF>\       Welcome to VAX/VMS UETP Version \
20 53 4D 56 2F 58 41 56 20  6F 74 20    0014
6E 6F 69 73 72 65 56 20 50  54 45 55    0020
                        20              002C
                                002D   625 VERSION:
                    00000035    002D   626          .BLKB   8
                       0A 0D    0035   627          .ASCII  <CR><LF>
                    0000002F    0037   628          WELCOML = .-WELCOME-8
                                0037   629
                                0037   630 TTCHAN:                                    ; Channel for the terminal
                        0000    0037   631          .WORD   0
                                0039   632
                                0039   633 ERROR_COUNT:                               ; Error count
                    00000000    0039   634          .LONG   0
                                003D   635
                                003D   636 FAO_BUF:                                  ; FAO output string descriptor
                    0000 012C   003D   637          .WORD   TEXT_BUFFER,0
                     0000004D'  0041   638          .ADDRESS BUFFER
                                0045   639
                                0045   640 BUFFER_PTR:                               ; Fake .ASCID buffer for misc. strings
                    0000 012C   0045   641          .WORD   TEXT_BUFFER,0           ; A word for length, a word for desc.
                     0000004D'  0049   642          .ADDRESS BUFFER
                                004D   643
                                004D   644 BUFFER:                                   ; FAO output and other misc. buffer
                    00000179    004D   645          .BLKB   TEXT_BUFFER
                                0179   646
                                0179   647 PASS_MSG:                                ; Used in startup msg
20 4C 55 21 20 2C 00000181'010E0000'    0179   648          .ASCID  \, !UL passes\
                  73 65 73 73 61 70     0187
                                018D   649
                                018D   650 LOAD_MSG:                                ; More for startup msg
20 4C 55 21 20 2C 00000195'010E0000'    018D   651          .ASCID  \, !UL load!%S\
                  53 25 21 64 61 6F 6C  019B
                                01A2   652
                                01A2   653 PARAM_MSG:                               ; Here is where the parameter portion
                    0000 0000   01A2   654          .WORD   0,0                     ; ...of the startup msg gets assembled
                     000001AA'  01A6   655          .ADDRESS PARAM_BUF
                                01AA   656 PARAM_BUF:
                    000002D6    01AA   657          .BLKB   TEXT_BUFFER
                                02D6   658
                                02D6   659 LOADS_DESC:                              ; Loads general purpose desc.
                    00000000    02D6   660          .LONG   0
                    00000000'   02DA   661          .ADDRESS 0
                                02DE   662
                                02DE   663 CPU_SCALE_DES:                           ; Descriptor for CPU scale value
      20 20 20 20 000002E6'010E0000'    02DE   664          .ASCID  /    /
                                02EA   665
                                02EA   666 WS_INUSE_DES:                            ; Descriptor for percent of WS in use
      20 20 20 20 000002F2'010E0000'    02EA   667          .ASCID  /    /
                                02F6   668
                                02F6   669 WS_INUSE:                                ; Storage for percent of WS in use
```

```
                CCCD3F4C  02F6   670          .LONG   PER_WS_INUSE            ; This is a floating point constant
                          02FA   671
                          02FA   672 DISK_BUFFER:                             ; System disk name
                000000FF  02FA   673          .LONG   LOGNAM_SIZE
                00000302' 02FE   674          .ADDRESS .+4
                00000401  0302   675          .BLKB   LOGNAM_SIZE
                          0401   676
                          0401   677 DEVBUF:                                  ; Gets device class of SYS$COMMAND...
                00000405  0401   678          .BLKL   1                       ; ...from $GETDVI
                          0405   679
                          0405   680 MSG_BLOCK:                               ; Auxiliary $GETMSG info
                00000409  0405   681          .BLKB   4
                          0409   682
                          0409   683 PAGE_COUNT:                             ; Floating point format memory page count
                00000000  0409   684          .FLOAT  0
                          040D   685 PAGE_BUF:                               ; String storage for memory size
                00000005  040D   686          .LONG   5
                00000415' 0411   687          .ADDRESS .+4
                0000041A  0415   688          .BLKB   5
                          041A   689
                          041A   690 QUAD_STATUS:                            ; IOSB for misc. system services
      00000000  00000000  041A   691          .QUAD   0
                          0422   692
                          0422   693 STATUS:                                 ; Status value on program exit
                00000000  0422   694          .LONG   0
                          0426   695
                          0426   696 EXIT_DESC:                              ; Exit handler descriptor
                00000000  0426   697          .LONG   0
                00000D7A' 042A   698          .ADDRESS EXIT_HANDLER
                00000001  042E   699          .LONG   1
                00000422' 0432   700          .ADDRESS STATUS
                          0436   701
                          0436   702 ARG_COUNT:                              ; Argument counter used by ERROR_EXIT
                00000000  0436   703          .LONG   0
                          043A   704
                          043A   705 FLAGS:                                  ; Miscellaneous flags.
                      00  043A   706          .BYTE   0                       ; See Equated Symbols for definitions
                          043B   707
                          043B   708 SYM_VAL_TABLE:                          ; Buffers for parameters P1-P4
                          043B   709
                          043B   710 P1_DESC:
                00000000  043B   711          .LONG   0
                0000045B' 043F   712          .ADDRESS P1_BUF
                          0443   713 P2_DESC:
                00000000  0443   714          .LONG   0
                0000055A' 0447   715          .ADDRESS P2_BUF
                          044B   716 P3_DESC:
                00000000  044B   717          .LONG   0
                00000659' 044F   718          .ADDRESS P3_BUF
                          0453   719 P4_DESC:
                00000000  0453   720          .LONG   0
                00000758' 0457   721          .ADDRESS P4_BUF
                          045B   722 P1_BUF:
                0000055A  045B   723          .BLKB   MAXSYM_SZ
                          055A   724 P2_BUF:
                00000659  055A   725          .BLKB   MAXSYM_SZ
                          0659   726 P3_BUF:
```

```
00000758  0659   727                .BLKB   MAXSYM_SZ
          0758   728 P4_BUF:
00000857  0758   729                .BLKB   MAXSYM_SZ
          0857   730
          0857   731
          0857   732 ANSWER:                                   ; Answer buffer desc
0000012C  0857   733                .LONG   TEXT_BUFFER
0000085F' 085B   734                .ADDRESS .+4
00000988  085F   735                .BLKB   TEXT_BUFFER
          0988   736
          0988   737 OUTLEN:                                   ; Output string desc
00000000  0988   738                .LONG   0
0000085F' 098F   739                .ADDRESS ANSWER+8
          0993   740
          0993   741 CPU_SCALE:                                ; This CPU's scale factor
00000000  0993   742                .FLOAT  0.0
          0997   743 PASS_COUNT:                               ; Total pass count
00000000  0997   744                .LONG   0
          0998   745 LOAD_COUNT:                               ; Total load count
00000000  0998   746                .LONG   0
          099F   747
          099F   748 VECTOR:                                   ; Message vector for $PUTMSG
     0003 099F   749                .WORD   3                  ; Arg count - total number of longwords
     0001 09A1   750                .WORD   ^B0001             ; Message flag
00741130  09A3   751                .LONG   UETP$_TEXT         ; Message ID
     0001 09A7   752                .WORD   1                  ; FAO arg count
     0000 09A9   753                .WORD   0                  ; New message flags
          09AB   754 MSG_DESC:
00000045' 09AB   755                .LONG   BUFFER_PTR         ; Address of message descriptor
          09AF   756
          09AF   757 WS_SIZE:                                  ; GETJPI results list
  00000000 09AF   758                .LONG   0
          09B3   759 JPI_ASTLM:
  00000000 09B3   760                .LONG   0
          09B7   761 JPI_BIOLM:
  00000000 09B7   762                .LONG   0
          09BB   763 JPI_BYTLM:
  00000000 09BB   764                .LONG   0
          09BF   765 JPI_CPULIM:
  00000000 09BF   766                .LONG   0
          09C3   767 JPI_ENQLM:
  00000000 09C3   768                .LONG   0
          09C7   769 JPI_DIOLM:
  00000000 09C7   770                .LONG   0
          09CB   771 JPI_FILLM:
  00000000 09CB   772                .LONG   0
          09CF   773 JPI_PGFLQUOTA:
  00000000 09CF   774                .LONG   0
          09D3   775 JPI_PRCLM:
  00000000 09D3   776                .LONG   0
          09D7   777 JPI_TQLM:
  00000000 09D7   778                .LONG   0
          09DB   779 PRIVS:
00000000 00000000 09DB   780                .QUAD   0
          09E3   781
          09E3   782 MEM_SIZE:                                 ; Total physical memory size in pages
  00000000 09E3   783                .LONG   0
```

```
                    09E7   784
                    09E7   785 MEM_FREE:
00000000            09E7   786         .LONG   0                       ; Physical memory not being used now
                    09EB   787
                    09EB   788 MEM_MODIFY:
00000000            09EB   789         .LONG   0                       ; Physical memory on the modified list
                    09EF   790
                    09EF   791 SWAP_SIZE:
00000000            09EF   792         .LONG   0                       ; Count of free process entry slots
                    09F3   793
                    09F3   794 PAGE_SIZE:
00000000            09F3   795         .LONG   0                       ; Secondary storage for paging in pages
                    09F7   796
                    09F7   797 SID:
000009FB            09F7   798         .BLKL   1                       ; $GETSYI returns SID register here
                    09FB   799
                    09FB   800 ;
                    09FB   801 ; CLI call back request descriptor
                    09FB   802 ;
                    09FB   803 CLI_REQ_DESC:
          05        09FB   804         .BYTE   CLI$K_CLISERV
        000A        09FC   805         .WORD   CLI$K_GETSYM                ; Get local sym is what we want to do
          01        09FE   806         .BYTE   CLI$K_LOCAL_SYM
00000000 00000000   09FF   807         .QUAD   0                       ; Desc of symbol name - CLI$Q_NAMDESC
00000000 00000000   0A07   808         .QUAD   0                       ; Desc of returned value -CLI$Q_VALDESC
```

```
                        .SBTTL  RMS-32 Data Structures
                        .ALIGN  LONG

                LOG_FAB:                                        ; Log file FAB
                        $FAB    FNM = <UETP.LOG>,-
                                RAT = CR,-
                                FAC = PUT
                LOG_RAB:                                        ; Log file RAB
                        $RAB    FAB = LOG_FAB,-
                                RBF = BUFFER,-
                                RSZ = TEXT_BUFFER
```

```
                                0AA4    822          .SBTTL  Main Program
                          00000000      823          .PSECT  UETINITOO,EXE,NOWRT,PAGE
                                0000    824
                                0000    825          .DEFAULT DISPLACEMENT,WORD
                                0000    826
                                0000    827 ;+
                                0000    828 ;       UETINITOO queries the user for UETP run-time information and welcomes
                                0000    829 ;       the user to UETP. The user is told what CPU type, memory configuration,
                                0000    830 ;       and system disk type he/she is running on. The user is prompted for the
                                0000    831 ;       number of complete passes he/she wants and if he/she responds with a
                                0000    832 ;       carriage return the default is one pass. The user is prompted for the
                                0000    833 ;       number of parallel simulated users that he wishes to have used in the
                                0000    834 ;       load test portion of the UETP. If he/she responds with a carriage return
                                0000    835 ;       UETINITOO calculates an appropriate value for the configuration that is
                                0000    836 ;       being used and informs the user as to what that value is. The user is
                                0000    837 ;       prompted for the report format (long or short) that is desired. If a
                                0000    838 ;       carriage return is the response, then long report format is used.
                                0000    839 ;       The UETP.LOG file is first created in this program as well.  The user
                                0000    840 ;       is allowed to choose to run the entire UETP or a subset of its phases,
                                0000    841 ;       with the default being the entire UETP.
                                0000    842 ;-
                                0000    843
                        0000    0000    844 .ENTRY UETINITOO,^M<>                           ; Entry mask
                                0002    845
                    6D  0B5B'CF  DE     0002    846          MOVAL   $SSERROR,(FP)          ; Declare exception handler
                                0007    847          $SETSFM_S ENBFLG = #1              ; Enable system service failure mode
                                0010    848          $DCLEXH_S DESBLK = EXIT_DESC       ; Declare an exit handler
                                001B    849
                                001B    850          $CREATE FAB = LOG_FAB,-            ; Create the log file
                                001B    851                  ERR = RMS_ERROR
                                002A    852          $CONNECT RAB = LOG_RAB,-
                                002A    853                  ERR = RMS_ERROR
                                0039    854
    002D'CF  00000000'GF  7D    0039    855          MOVQ    G^SYS$GQ_VERSION,VERSION ; Get the system version number
      09AB'CF    0000'CF  DE    0042    856          MOVAL   WELCOME,MSG_DESC           ; Message desc
                                0049    857          $PUTMSG_S MSGVEC = VECTOR,-        ; Go ahead and output msg
                                0049    858                    ACTRTN = ACTRTN          ; Output it to log file as well
                                005C    859          $SETPRN_S PRCNAM = TEST_NAME       ; Set the process name
                                0067    860
                                0067    861          $GETJPI_S ITMLST = USER_LIST       ; Get the username, privs and quotas
      0008'CF    0000'CF  29    007C    862          CMPC3   ACNT_NAME,ACNT_NAME+8,-    ; Are we in the right account?
                  004D'CF                 0083    863                  BUFFER
                        1A  13  0086    864          BEQL    10$                        ; BR if no...
                  01E3'CF  DF   0088    865          PUSHAL  WRONG_ACCOUNT              ; ...else report and exit
                        01  DD  008C    866          PUSHL   #1                         ; Arg count
                00741132  8F   DD  008E    867          PUSHL   #UETP$_TEXT!STS$K_ERROR    ; Signal name
                        03  DD  0094    868          PUSHL   #3                         ; Parameter count
    0422'CF  00000000'8F  D0    0096    869          MOVL    #SS$_BADPARAM,STATUS       ; Set the exit status
                    0C7C  31    009F    870          BRW     ERROR_EXIT                 ; Give the user the last rights
                                00A2    871 10$:
                                00A2    872          $GETDVI_S DEVNAM = SYS$COMMAND,-   ; Get the name of...
                                00A2    873                    IOSB   = QUAD_STATUS,-   ; ...device which may abort test
                                00A2    874                    ITMLST = COMMAND_ITMLST
                      37  041A'CF  E8   00BE    875          BLBS    QUAD_STATUS,20$            ; BR if all went OK
                    52    041A'CF  3C   00C3    876          MOVZWL  QUAD_STATUS,R2             ; We had a problem. Extract error code
                                00C8    877          $GETMSG_S MSGID = R2,-             ; Get message text associated with error
                                00C8    878                    MSGLEN = BUFFER_PTR,-
```

```
                     00C8   879                 BUFADR = FAO_BUF
          0045'CF    7F  00DD   880         PUSHAQ  BUFFER_PTR               ; Let user know what went wrong...
              01     DD  00E1   881         PUSHL   #1
      00741132 8F    DD  00E3   882         PUSHL   #UETP$_TEXT!STS$K_ERROR
          01AF'CF    7F  00E9   883         PUSHAQ  COMMAND_DVI_FAILED
              01     DD  00ED   884         PUSHL   #1
      00741132 8F    DD  00EF   885         PUSHL   #UETP$_TEXT!STS$K_ERROR
              06     DD  00F5   886         PUSHL   #6
            0C24     31  00F7   887         BRW     ERROR_EXIT              ; ...and bail out
  0401'CF   00'8F    91  00FA   888 20$:     CMPB    #DC$_TERM,DEVBUF        ; Were we invoked from a terminal?
              4A     12  0100   889         BNEQ    30$                     ; BR if not
        043A'CF      04  88  0102   890         BISB2   #TERMINALM,FLAGS        ; Set terminal flag
                         0107   891         $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C ASTs if we were
                         0107   892                   CHAN   = TTCHAN
                         0118   893         $QIOW_S CHAN     = TTCHAN,-     ; Enable CTRL/C ASTs...
                         0118   894                 FUNC     = #IO$_SETMODE!IO$M_CTRLCAST,-
                         0118   895                 P1       = CCASTHAND
          000F'CF    DF  0139   896         PUSHAL  TEST_NAME               ; ...and tell the user...
              01     DD  013D   897         PUSHL   #1
        0074832B 8F    DD  013F   898         PUSHL   #UETP$_ABORTC!STS$K_SUCCESS
  00000000'GF    03    FB  0145   899         CALLS   #3,G^LIB$SIGNAL         ; ...how to abort gracefully
                         014C   900 30$:
                         014C   901         $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                         014C   902                   RSLLEN = BUFFER_PTR,-
                         014C   903                   RSLBUF = FAO_BUF
          0045'CF    7F  0165   904         PUSHAQ  BUFFER_PTR              ; Convert to upper case
          0045'CF    7F  0169   905         PUSHAQ  BUFFER_PTR
  00000000'GF    02    FB  016D   906         CALLS   #2,G^STR$UPCASE
0030'DF   002C'CF    39  0174   907         MATCHC  DUMP,@DUMP+4,-          ; Are we to run in dump mode?
004D'CF   0045'CF        017B   908                 BUFFER_PTR,BUFFER
              05     12  0181   909         BNEQ    35$                     ; BR if not
        043A'CF      10  88  0183   910         BISB2   #DUMPM,FLAGS            ; Else set the flag bit
                         0188   911 35$:
```

```
                                                M 3
                     0188   913 ;+
                     0188   914 ;       Go through this process' privileges and quotas.  If something
                     0188   915 ;       nonstandard shows up, give a warning.
                     0188   916 ;-
                  52 D4   0188   917         CLRL    R2                      ; Init an index variable
                     018A   918
               56 02B3'CF DE   018A   919         MOVAL   PRV_STR,R6              ; List non-standard privs first
            54 0624'CF42 9A   018F   920 40$:     MOVZBL  OFFSET[R2],R4           ; Get the offset of the priv
53 09DB'CF 01 54 EF   0195   921         EXTZV   R4,#1,PRIV$,R3          ; Get the priv
            53 064B'CF42 D1   019C   922         CMPL    EXPECTED[R2],R3         ; Check it
                  3D 10   01A2   923         BSBB    80$                     ; Br if bad
               E7 52 1E   F2   01A4   924         AOBLSS  #PRIV_CNT,R2,40$        ; Do all privs
                     01A8   925
               56 02BD'CF DE   01A8   926         MOVAL   QUO_STR,R6              ; Now we're listing non-standard quotas
            54 0624'CF42 9A   01AD   927 60$:     MOVZBL  OFFSET[R2],R4           ; Get the offset of the quota
09B3'CF44 064B'CF42 D1   01B3   928         CMPL    EXPECTED[R2],JPI_ASTLM[R4] ; Check it
                  23 10   01BC   929         BSBB    80$                     ; Br if bad
               EB 52 27   F2   01BE   930         AOBLSS  #PRIV_CNT+QUOT_CNT,R2,60$ ; Do all quotas
                     01C2   931
            17 043A'CF 03   E5   01C2   932         BBCC    #PRIV_PRNTV,FLAGS,70$   ; Only print the ending message once
                  0250'CF DF   01C8   933         PUSHAL  ENDSTR                  ; push the message address
                  00010001 8F   DD   01CC   934         PUSHL   #^X10001                ; push the arg count
                  00741130 8F   DD   01D2   935         PUSHL   #UETP$_TEXT!STS$K_WARNING ; push the signal name
               00000000'GF 03   FB   01D8   936         CALLS   #3,G^LIB$SIGNAL         ; print the ending error message
                  69 11   01DF   937 70$:     BRB     110$
                     01E1   938
                     01E1   939 ; Subroutine to list non-standard privileges and quotas.
                  66 13   01E1   940 80$:     BEQL    100$                    ; Don't complain if priv/quota is OK
            2F 043A'CF 03   E2   01E3   941         BBSS    #PRIV_PRNTV,FLAGS,90$   ; Only print error message header once
                  0039'CF D6   01E9   942         INCL    ERROR_COUNT             ; Bump the error count
                  0238'CF DF   01ED   943         PUSHAL  STRSTR                  ; Push the string address
               000F0001 8F   DD   01F1   944         PUSHL   #^XF0001                ; Push the arg count
               00741130 8F   DD   01F7   945         PUSHL   #UETP$_TEXT!STS$K_WARNING ; Push the signal name
                  0039'CF DD   01FD   946         PUSHL   ERROR_COUNT             ; Finish off arg list...
                  000F'CF DF   0201   947         PUSHAL  TEST_NAME               ; ...
               00010002 8F   DD   0205   948         PUSHL   #^X10002                ; ...
               00748022 8F   DD   0208   949         PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
            00000000'GF 07   FB   0211   950         CALLS   #7,G^LIB$SIGNAL         ; Print the error message
                     0218   951 90$:
                     0218   952         $FAO_S  CTRSTR = CTRSTR,-
                     0218   953                 OUTLEN = BUFFER_PTR,-
                     0218   954                 OUTBUF = FAO_BUF,-
                     0218   955                 P1     = NAM_PTRS[R2],-
                     0218   956                 P2     = R6             ; Generate the string
                  0045'CF DF   0232   957         PUSHAL  BUFFER_PTR              ; Push the address...
               00010001 8F   DD   0236   958         PUSHL   #^X10001                ; ...the arg count...
               00741130 8F   DD   023C   959         PUSHL   #UETP$_TEXT!STS$K_WARNING ; ...the signal name...
            00000000'GF 03   FB   0242   960         CALLS   #3,G^LIB$SIGNAL         ; ...and print the message
                  05 0249   961 100$:    RSB                             ; Return for further checking
```

```
                                024A   963 ;+
                                024A   964 ;       Figure out the CPU type in preparation for defining the number of load
                                024A   965 ;       test loads to run.  In their wisdom, the engineers who designed
                                024A   966 ;       closely-coupled CPUs and "mid-life kicker" CPUs have chosen to keep
                                024A   967 ;       the same CPU type as the base CPU, but there are other hints to figure
                                024A   968 ;       out what one is running on.  Figuring whether we have a multiple CPU
                                024A   969 ;       configuration uses the VMS location EXE$GL_MP, which indicates the
                                024A   970 ;       presence of a tightly-coupled, second processor if non-zero.  SUPERSTAR
                                024A   971 ;       sets a bit in the SID register.  Note that we don't use the VMS macro,
                                024A   972 ;       CPUDISP.
                                024A   973 ;-
                                024A   974 110$:
                                024A   975         $GETSYI_S ITMLST = GETSYI_ITMLST ; Get misc system info
                    18   EF     025F   976         EXTZV   #PR$V_SID_TYPE,-        ; Figure the CPU type from that
58  09F7'CF        08           0261   977                 #PR$S_SID_TYPE,SID,R8
                09  58   3A     0266   978         LOCC    R8,#CTT_LENGTH,-        ; See if VMS knows of that CPU...
                    08AA'CF      0269   979                 CPU_TYPE_TABLE
    50  09  50      C3     026C   980         SUBL3   R0,#CTT_LENGTH,R0       ; ...and convert to that type's offset
    09  00  50      8F     0270   981         CASEB   R0,#0,#CTT_LENGTH       ; BR to set up for the correct CPU
                    0039'  0274   982 120$:           .WORD 140$-120$         ; Illegal or unknown CPU
                    001A'  0276   983                 .WORD 121$-120$         ; STAR-based CPUs
                    0039'  0278   984                 .WORD 140$-120$         ; COMET
                    0039'  027A   985                 .WORD 140$-120$         ; NEBULA
                    0039'  027C   986                 .WORD 140$-120$         ; VENUS
                    0039'  027E   987                 .WORD 140$-120$         ; SCORPIO
                    0039'  0280   988                 .WORD 140$-120$         ; NAUTILUS
                    0039'  0282   989                 .WORD 140$-120$         ; SEAHORSE I
                    0039'  0284   990                 .WORD 140$-120$         ; microVAX chip
                           0286   991         .REPEAT 4
                           0286   992         NOP                             ; Fudge so we can patch in new CPUs
                        01 0286   993         .ENDR
                58      D4 028A   994         CLRL    R8                      ; Default value - illegal CPU
                1F      11 028C   995         BRB     140$
                           028E   996 121$:
    00000000'GF         D5 028E   997         TSTL    G^EXE$GL_MP             ; Are we multiprocessing?
                03      13 0294   998         BEQL    122$                    ; BR to check SUPERSTAR if not
            58  01      CE 0296   999         MNEGL   #1,R8                   ; Use a different offset if we are
                           0299  1000 122$:
0E  09F7'CF     17      E1 0299  1001         BBC     #23,SID,140$            ; We're already correct if 11/780
            58  02      CE 029F  1002         MNEGL   #2,R8                   ; Set up SUPERSTAR offset
    00000000'GF         D5 02A2  1003         TSTL    G^EXE$GL_MP             ; Are we multiprocessing as well?
                03      13 02A8  1004         BEQL    140$                    ; BR to get scale & text if not
            58  03      CE 02AA  1005         MNEGL   #3,R8                   ; Use a different offset if we are
                           02AD  1006         BRB     140$                    ; Fall into default processing
                           02AD  1007 140$:
0993'CF     0941'CF48   50 02AD  1008         MOVF    CPU_SCALE_TABLE[R8],-   ; Save the CPU scale factor
                           02B5  1009                 CPU_SCALE
    58      08E5'CF48   D0 02B5  1010         MOVL    CPU_NAME_TABLE[R8],R8   ; Ah! that's what kind of CPU it is
                           02BB  1011         $CMEXEC_S ROUTIN = GET_MEM_INFO ; Figure various memory limits
                           02C8  1012         $FAO_S  CTRSTR = SYSTEM,-       ; Generate the string
                           02C8  1013                 OUTLEN = BUFFER_PTR,-
                           02C8  1014                 OUTBUF = FAO_BUF,-
                           02C8  1015                 P1     = R8,-
                           02C8  1016                 P2     = MEM_SIZE
09AB'CF     0045'CF     DE 02E1  1017         MOVAL   BUFFER_PTR,MSG_DESC
                           02E8  1018         $PUTMSG_S MSGVEC = VECTOR,-     ; Go ahead and output msg
                           02E8  1019                   ACTRTN = ACTRTN       ; Output it to log file as well
```

```
                     02FB  1020          $TRNLOG_S LOGNAM = SYSDISK,-     ; Get the system disk designation
                     02FB  1021                    RSLLEN = DISK_BUFFER,-
                     02FB  1022                    RSLBUF = DISK_BUFFER
                     0314  1023          $FAO_S  CTRSTR = DISK,-          ; Format system disk msg
                     0314  1024                  OUTLEN = BUFFER_PTR,-
                     0314  1025                  OUTBUF = FAO_BUF,-
                     0314  1026                  P1     = #DISK_BUFFER
09AB'CF  0045'CF  DE 032D  1027          MOVAL   BUFFER_PTR,MSG_DESC
                     0334  1028          $PUTMSG_S MSGVEC = VECTOR,-      ; Go ahead and output msg
                     0334  1029                    ACTRTN = ACTRTN        ; Output it to log file as well
```

```
                     0347  1031 ;+
                     0347  1032 ;
                     0347  1033 ;       Here we call the CLI to get values for local symbols P1-P4.  If they
                     0347  1034 ;       are not defined SYS$CLI returns LIB$_NOSUCHSYM and each associated
                     0347  1035 ;       descriptor is left with length zero.
                     0347  1036 ;-
                59   D4  0347  1037          CLRL    R9                              ; Symbols found counter
          56 09FB'CF   DE  0349  1038          MOVAL   CLI_REQ_DESC,R6                 ; CLI request block
          57 0A6E'CF   DE  034E  1039          MOVAL   SYM_NAM_TABLE,R7                ; Parameter names
          58 043B'CF   DE  0353  1040          MOVAL   SYM_VAL_TABLE,R8                ; Table for returned values
             5B   04   D0  0358  1041          MOVL    #SYMBOL_CNT,R11                 ; Loop count
                     035B  1042 150$:
          04 A6   87   7D  035B  1043          MOVQ    (R7)+,CLI$Q_NAMDESC(R6) ; Put symbol name desc in req block
                0C A6   7C  035F  1044          CLRQ    CLI$Q_VALDESC(R6)               ; Init return desc
                   66   DF  0362  1045          PUSHAL  (R6)                            ; Push address of the req block
00000000'GF        01   FB  0364  1046          CALLS   #1,G^SYS$CLI                    ; Callback to the CLI
00000000'8F        50   D1  036B  1047          CMPL    R0,#SS$_NORMAL                  ; Did we find it
                   1D   12  0372  1048          BNEQ    160$                            ; BR if not
                0C A6   B5  0374  1049          TSTW    CLI$Q_VALDESC(R6)               ; Test for zero length
                   18   13  0377  1050          BEQL    160$                            ; Br if zero length
          0C A6    20   3B  0379  1051          SKPC    #^A/ /,CLI$Q_VALDESC(R6),- ; Make sure we did not get all spaces
                10 B6       037D  1052                  @<CLI$Q_VALDESC+4>(R6)
                   10   13  037F  1053          BEQL    160$                            ; BR if only spaces
                   59   D6  0381  1054          INCL    R9                              ; Count this one found
          8B    0C A6   D0  0383  1055          MOVL    CLI$Q_VALDESC(R6),(R8)+ ; Save return length
                0C A6   28  0387  1056          MOVC3   CLI$Q_VALDESC(R6),-     ; ...and value
             00 B8   10 B6       038A  1057                  @<CLI$Q_VALDESC+4>(R6),@(R8)
             5B   04   C2  038E  1058          SUBL2   #4,R8                           ; Reset R8 to start of present descriptor
                     0391  1059 160$:
             58   08   C0  0391  1060          ADDL2   #8,R8                   ; Move PTR to next value descriptor
               C4 5B   F5  0394  1061          SOBGTR  R11,150$                ; Repeat until we tried them all
                   59   D5  0397  1062          TSTL    R9                      ; Were any symbols defined?
                   0B   12  0399  1063          BNEQ    PHASE                   ; BR if we found at least one
       05 043A'CF   02   E1  039B  1064          BBC     #TERMINALV,FLAGS,PHASE  ; BR if we are not connected to a
                     03A1  1065                                                  ;  terminal -we will use default values
          043A'CF   02   88  03A1  1066          BISB2   #PROMPTM,FLAGS          ; No parameters were defined and we are
                     03A6  1067                                                  ;  connected to a terminal so set the
                     03A6  1068                                                  ;  flag for prompting
```

```
                     03A6  1069 ;+
                     03A6  1070 ;       See if the user wants to run the entire UETP or a subset o. its phases.
                     03A6  1071 ;       Define a logical name, UETPPHASE, with the result.  UETP.COM will use
                     03A6  1072 ;       the translation of UETPPHASE to execute the appropriate phase(s).
                     03A6  1073 ;-
                     03A6  1074
                     03A6  1075 PHASE:
      31 043A'CF  01 E0  03A6  1076          BBS     #PROMPTV,FLAGS,10$          ; BR if we need to prompt
098B'CF  043B'CF   B0  03AC  1077          MOVW    P1_DESC,OUTLEN              ; No prompting, phase is P1 param...
                7F 13  03B3  1078          BEQL    40$                         ; ...but supply default if null
           043B'CF 7F  03B5  1079          PUSHAQ  P1_DESC                     ; We have some request, so...
           043B'CF 7F  03B9  1080          PUSHAQ  P1_DESC                     ; ...for matching's sake...
 00000000'GF   02 FB  03BD  1081          CALLS   #2,G^STR$UPCASE             ; ...get it all uppercase
045B'CF  043B'CF 29  03C4  1082          CMPC3   P1_DESC,P1_BUF,-            ; Did user specifically request...
           0B45'DF            03CB  1083                  @KEY_ALL_DESC+4     ; ...to run all phases?
                64 13  03CE  1084          BEQL    40$                         ; BR if so
098F'DF  045B'CF  043B'CF 28  03D0  1085          MOVC3   P1_DESC,P1_BUF,@OUTLEN+4 ; Use user's reply since there is one
                0110 31  03DA  1086          BRW     200$                        ; Join code which has user's selection
                     03DD  1087 10$:
           0988'CF  3F  03DD  1088          PUSHAW  OUTLEN                      ; Get user's choice - reply length...
           0A96'CF  7F  03E1  1089          PUSHAQ  PHASE_PROMPT                ; ...prompt string...
           0857'CF  7F  03E5  1090          PUSHAQ  ANSWER                      ; ...reply string...
 00000000'GF   03 FB  03E9  1091          CALLS   #3,G^LIB$GET_COMMAND        ; ...- for the phase(s) to execute
              08 50  E8  03F0  1092          BLBS    R0,20$                      ; BR if we could read response
        0422'CF  50  D0  03F3  1093          MOVL    R0,STATUS                   ; Use error code as exit status
              095C 31  03F8  1094          BRW     FINI
           098B'CF  B5  03FB  1095 20$:    TSTW    OUTLEN                      ; Was there some explicit request?
                33 13  03FF  1096          BEQL    40$                         ; BR if not - supply default
           098B'CF  7F  0401  1097          PUSHAQ  OUTLEN                      ; We have some request, so...
           098B'CF  7F  0405  1098          PUSHAQ  OUTLEN                      ; ...for matching's sake...
 00000000'GF   02 FB  0409  1099          CALLS   #2,G^STR$UPCASE             ; ...get it all uppercase
           004D'CF  DF  0410  1100          PUSHAL  BUFFER                      ; This will tell which reply we got...
           0B2D'CF  DF  0414  1101          PUSHAL  SELECT_PHASE                ; ...this tells the possibilities...
           098B'CF  7F  0418  1102          PUSHAQ  OUTLEN                      ; ...and this is the text of the reply
00000000'GF   03 FB  041C  1103          CALLS   #3,G^LIB$LOOKUP_KEY         ; See if we want all or a subset
           50   00' D1  0423  1104          CMPL    S^#SS$_NORMAL,R0            ; Did we find a reasonable reply?
                06 13  0426  1105          BEQL    30$                         ; BR if we did
              08A8 30  0428  1106          BSBW    SYNTAX_ERROR                ; Complain if we did not...
              FF78 31  042B  1107          BRW     PHASE                       ; ...and start all over
           004D'CF  D5  042E  1108 30$:    TSTL    BUFFER                      ; Was a subset requested (add'l prompt)?
                32 12  0432  1109          BNEQ    100$                        ; Yes, go do second prompt
                       0434  1110          ; The user requested all phases.  Fall into that code.
                       0434  1111
                       0434  1112 40$:    ; The user wants all UETP phases, either explicitly or implicitly.
   56   0B5C'CF  DE  0434  1113          MOVAL   PHASE_TABLE,R6              ; Get the list of phase names...
57   86    FF 8F  78  0439  1114          ASHL    #-1,(R6)+,R7                ; ...their count...
           01A2'CF  B4  043E  1115          CLRW    PARAM_MSG                   ; ...an accumulator for total length...
   53   01AA'CF  DE  0442  1116          MOVAL   PARAM_BUF,R3                ; ...a place to concatenate them...
        55   66  D0  0447  1117 50$:    MOVL    (R6),R5                     ; ...the pointer to a name...
        58   65  9B  044A  1118          MOVZBW  (R5),R8                     ; ...the length of an individual name...
63   01 A5   58  28  044D  1119          MOVC3   R8,1(R5),(R3)               ; ...the text forming the name...
   01A2'CF   58  A0  0452  1120          ADDW2   R8,PARAM_MSG
           83   20  90  0457  1121          MOVB    #^A/ /,(R3)+                ; ...and a separator between names...
           01A2'CF  B6  045A  1122          INCW    PARAM_MSG
                86  73  045E  1123          TSTD    (R6)+
             E4 57  F5  0460  1124          SOBGTR  R7,50$                      ; ...to form the default of all names
              0125  31  0463  1125          BRW     300$                        ; Go process the default list
```

```
                                0466  1127 ;
                                0466  1128 ; Form on the stack an $FAOL PRMLST of UETP phase names, based on the list to be
                                0466  1129 ; passed to LIB$LOOKUP_KEY.  Be somewhat clever in listing the names, inserting
                                0466  1130 ; proper spacing and new lines.  In doing so, remember that $FAOL uses a FIFO
                                0466  1131 ; algorithm for removing items from the PRMLST.  We'll preallocate a worst case
                                0466  1132 ; amount of space on the stack (which is normally LIFO!) and stick pointers to
                                0466  1133 ; .ASCIC strings on the list in FIFO order. The space needed takes into account
                                0466  1134 ; that we could need three longwords per phase name (the name, separator
                                0466  1135 ; characters and newline), that the list of names has a count of longwords at
                                0466  1136 ; its front instead of a count of entries, and that we're allocating bytes, not
                                0466  1137 ; longwords.  Use the $FAOL results as the prompt for the phase we want to
                                0466  1138 ; execute.
                                0466  1139 ;
                                0466  1140 100$:
                56    5E    D0  0466  1141          MOVL    SP,R6                   ; R6 will clean up the stack later
                      57    D4  0469  1142          CLRL    R7                      ; R7 counts the .ASCIC strings
             58   0B5C'CF    DE  046B  1143          MOVAL   PHASE_TABLE,R8          ; R8 points to the phase name list
             59   68    06    C5  0470  1144          MULL3   #6,(R8),R9              ; Figure worst case of space we'll need
                  5E    59    C2  0474  1145          SUBL2   R9,SP                   ; Preallocate space on the stack
                  59    5E    D0  0477  1146          MOVL    SP,R9                   ; R9 points to base of FIFO list
          5A  88    FF  8F    78  047A  1147          ASHL    #-1,(R8)+,R10           ; R10 counts phase names remaining
                                047F  1148                                          ; (R8 now points to ptr to first name)
                5B    08    D0  047F  1149          MOVL    #8,R11                  ; R11 counts characters on a line
                                0482  1150                                          ; (The listing of phases starts one...
                                0482  1151                                          ; ...tab stop from the left margin)
                                0482  1152 110$:
          55  5B    00  B8    81  0482  1153          ADDB3   @(R8),R11,R5            ; If phase name + current line width...
    000A  55  0ACC'CF   50  8F    9D  0487  1154          ACBB    #80,COMMA_BLANK,R5,120$ ; ...+ separator chars .GT. 80...
             89   0ACF'CF    DE  0490  1155          MOVAL   NEW_LINE,(R9)+          ; ...then start a new line,...
                  55    08    D0  0495  1156          MOVL    #8,R5                   ; ...figure what column we're on,...
                        57    D6  0498  1157          INCL    R7                      ; ...and count another .ASCIC string
                                049A  1158 120$:
                89    88    D0  049A  1159          MOVL    (R8)+,(R9)+             ; Put a phase name on $FAOL PRMLST
                        88    D5  049D  1160          TSTL    (R8)+                   ; Skip over LIB$LOOKUP_KEY assoc. value
             89 0ACC'CF         DE  049F  1161          MOVAL   COMMA_BLANK,(R9)+       ; Put separator chars on $FAOL PRMLST
                  57    02    C0  04A4  1162          ADDL2   #2,R7                   ; Count the .ASCIC strings we've pushed
                  5B    55    D0  04A7  1163          MOVL    R5,R11                  ; Update current line width
                    D5  5A    F5  04AA  1164          SOBGTR  R10,110$                ; Loop if there are more phase names
             79 0B1E'CF         DE  04AD  1165          MOVAL   WHICH_PHASE2,-(R9)      ; Use second half of prompt to...
                                04B2  1166                                          ; ...overwrite trailing separator chars
                        57    DD  04B2  1167          PUSHL   R7                      ; Put .ASCIC count in front of PRMLST
                58    5E    D0  04B4  1168          MOVL    SP,R8                   ; Save pointer to the PRMLST
                                04B7  1169          $FAOL_S CTRSTR = WHICH_PHASE1,- ; Form prompt for...
                                04B7  1170                  OUTBUF = FAO_BUF,-
                                04B7  1171                  OUTLEN = BUFFER_PTR,-
                                04B7  1172                  PRMLST = (R8)
                5E    56    D0  04CC  1173          MOVL    R6,SP                   ; (Restore stack: rid it of PRMLST)
                0988'CF         DF  04CF  1174          PUSHAL  OUTLEN
                0045'CF         DF  04D3  1175          PUSHAL  BUFFER_PTR
                0857'CF         DF  04D7  1176          PUSHAL  ANSWER
00000000'GF       03    FB    04DB  1177          CALLS   #3,G^LIB$GET_COMMAND    ; ...deciding which phase to run
                08  50    E8  04E2  1178          BLBS    R0,200$                 ; Can we read SYS$COMMAND?
          0422'CF   50    D0  04E5  1179          MOVL    R0,STATUS               ; Supply an exit status...
                  086A  31    04EA  1180          BRW     FINI                    ; ...and bail out if we can't
```

```
                                04ED  1182 ;
                                04ED  1183 ; Now that we've got a (list of) phase name(s) from P1 or prompt, see if it
                                04ED  1184 ; (they) is (are) valid.  P1_DESC can be scratch.  Accumulate in PARAM_MSG.
                                04ED  1185 ;
            098B'CF    7F   04ED  1186 200$:   PUSHAQ  OUTLEN                  ; Convert possible...
            098B'CF    7F   04F1  1187         PUSHAQ  OUTLEN                  ; ...lowercase answer...
  00000000'GF      02   FB  04F5  1188         CALLS   #2,G^STR$UPCASE         ; ...to uppercase
            52       09   D0  04FC  1189         MOVL    #^A/    /,R2            ; We'll want a list containing only...
                     7A   10  04FF  1190         BSBB    220$                    ; ...blanks and phase names...
            52       2C   D0  0501  1191         MOVL    #^A/,/,R2               ; ...so convert other separators...
                     75   10  0504  1192         BSBB    220$                    ; ...to blanks
         56  098B'CF    7D   0506  1193         MOVQ    OUTLEN,R6               ; Prime pump to form desc for first...
                              0508  1194                                         ; ...possible phase name
            01A2'CF    B4   0508  1195         CLRW    PARAM_MSG               ; We have no phase names accepted yet...
         53 01A6'CF    D0   050F  1196         MOVL    PARAM_MSG+4,R3          ; ...but when we do, they're copied here
            098B'CF    B5   0514  1197         TSTW    OUTLEN                  ; Special case: have we an empty list?
                     0D   12   0518  1198         BNEQ    210$                    ; BR if not, we can parse it
            043B'CF    B4   051A  1199         CLRW    P1_DESC                 ; Set up to call our error routine...
     50   00741108   8F   D0   051E  1200         MOVL    #UETP$_BADKEY,R0        ; ...
                     7A   11   0525  1201         BRB     400$                    ; ...and complain
                              0527  1202 210$:
         67  56      20   3B   0527  1203         SKPC    #^A/ /,R6,(R7)          ; Pass over leading/intervening blanks
                     5E   13   052B  1204         BEQL    300$                    ; BR if no possible phase names left
            043B'CF  50   7D   052D  1205         MOVQ    R0,P1_DESC              ; Save desc for possible phase name
         61  50      20   3A   0532  1206         LOCC    #^A/ 7,R0,(R1)          ; Find end of the possible phase name
            043B'CF  50   C2   0536  1207         SUBL2   R0,P1_DESC              ; Now get the true length of the name
                     56   50   7D   053B  1208         MOVQ    R0,R6                   ; Set up pointers for the next name
   0045'CF     012C  8F   3C   053E  1209         MOVZWL  #TEXT_BUFFER,BUFFER_PTR
            0045'CF    DF   0545  1210         PUSHAL  BUFFER_PTR              ; See which phase: out-len...
            0045'CF    DF   0549  1211         PUSHAL  BUFFER_PTR              ; ...full-dsc-adr...
                     00   DD   054D  1212         PUSHL   #0                      ; ...key-value-adr...
            0B5C'CF    DF   054F  1213         PUSHAL  PHASE_TABLE             ; ...key-table-adr...
            043B'CF    DF   0553  1214         PUSHAL  P1_DESC                 ; ...str-dsc-adr...
  00000000'GF      05   FB  0557  1215         CALLS   #5,G^LIB$LOOKUP_KEY     ; ...
                     50   00'  B1   055E  1216         CMPW    S^#SS$_NORMAL,R0        ; Did we get a unique match?
                     3E   12   0561  1217         BNEQ    400$                    ; BR if not - go to our error routine
63  004D'CF    0045'CF    28   0563  1218         MOVC3   BUFFER_PTR,BUFFER,(R3)  ; Copy an unabbreviated phase name
   01A2'CF     0045'CF    A0   0568  1219         ADDW2   BUFFER_PTR,PARAM_MSG    ; Include its length in the descriptor
            83       20   90   0572  1220         MOVB    #^A/ /,(R3)+            ; Separate phase names...
            01A2'CF    B6   0575  1221         INCW    PARAM_MSG               ; ...and count the separators, too
                     AC   11   0579  1222         BRB     210$                    ; Loop for another phase name
                              057B  1223
098F'DF     098B'CF    52   3A   057B  1224 220$:   LOCC    R2,OUTLEN,@OUTLEN+4     ; Find a separator we want to convert
                     05   13   0583  1225         BEQL    230$                    ; BR if none are left
            61       20   90   0585  1226         MOVB    #^A/ /,(R1)             ; Convert it to a blank...
                     F1   11   0588  1227         BRB     220$                    ; ...and look for another
                     05   058A  1228 230$:   RSB
                              058B  1229
                              058B  1230 ;
                              058B  1231 ; We've got our phase name list.  We define a group logical name so that it
                              058B  1232 ; will persist beyond running this image.  Note that PARAM_MSG and PARAM_BUF
                              058B  1233 ; are preserved for the FINAL_MESSAGE routine.
                              058B  1234 ;
                              058B  1235 300$:   $CRELOG_S LOGNAM = UETPPHASE,-  ; Define logical name for UETP.COM label
                              058B  1236                   EQLNAM = PARAM_MSG,-
                              058B  1237                   TBLFLG = #1            ; It's a group logical name
                0085     31   059E  1238         BRW     PASS                    ; Process the next question
```

```
                          05A1  1240 ;
                          05A1  1241 ; We were passed a bum phase name.  That's not too bad if we're interactive,
                          05A1  1242 ; (just reprompt) but give up if we're not interactive (we were passed a bad
                          05A1  1243 ; parameter).
                          05A1  1244 ;
                          05A1  1245 400$:
02 043A'CF    01   E0     05A1  1246          BBS     #PROMPTV,FLAGS,410$      ; BR if we are prompting because...
                00'   DD  05A7  1247          PUSHL   S^#SS$_BADPARAM          ; ...if not we'll want add'l message
                          05A9  1248 410$:
                01   BB   05A9  1249          PUSHR   #^M<R0>                  ; Save LOOKUP_KEY status over $GETMSG
                          05AB  1250          $GETMSG_S MSGID = R0,-           ; Figure out...
                          05AB  1251                    MSGLEN = BUFFER_PTR,-  ; ...if the message...
                          05AB  1252                    BUFADR = FAO_BUF,-     ; ...associated with our fail code...
                          05AB  1253                    FLAGS  = #0,-          ; ...needs any $FAO args
                          05AB  1254                    OUTADR = MSG_BLOCK
                01   BA   05C2  1255          POPR    #^M<R0>                  ; Restore failure code
    52    0406'CF   9A   05C4  1256          MOVZBL  MSG_BLOCK+1,R2           ; Make $FAO arg count more useable
                09   13   05C9  1257          BEQL    420$                     ; BR if there are no associated args
          043B'CF   DF   05CB  1258          PUSHAL  P1_DESC                  ; Assume that the arg is the bad string
                01   DD   05CF  1259          PUSHL   #1
                52   02   D0   05D1  1260          MOVL    #2,R2
                          05D4  1261 420$:
                50   DD   05D4  1262          PUSHL   R0                       ; Yell at the user if bad reply
                          05D6  1263          $FAO_S  CTRSTR = INVALID_PHASE_MSG,-
                          05D6  1264                  OUTLEN = BUFFER_PTR,-
                          05D6  1265                  OUTBUF = FAO_BUF,-
                          05D6  1266                  P1     = #P1_DESC
          0045'CF   DF   05EF  1267          PUSHAL  BUFFER_PTR
                01   DD   05F3  1268          PUSHL   #1
       00741132   8F   DD   05F5  1269          PUSHL   #UETP$_TEXT!STS$K_ERROR
 19 043A'CF       01   E1   05FB  1270          BBC     #PROMPTV,FLAGS,430$      ; BR if not prompting
          00D9'CF   DF   0601  1271          PUSHAL  SYNTAX_ERROR_MSG
                01   DD   0605  1272          PUSHL   #1
       00741132   8F   DD   0607  1273          PUSHL   #UETP$_TEXT!STS$K_ERROR
          52       07   C0   060D  1274          ADDL2   #7,R2                    ; Add to old count for LIB$SIGNAL args
00000000'GF        52   FB   0610  1275          CALLS   R2,G^LIB$SIGNAL
              FE4C      31   0617  1276          BRW     100$                     ; Politely ask again
                          061A  1277 430$:
    7E    52    05   C1   061A  1278          ADDL3   #5,R2,-(SP)              ; Add to old count for ERROR_EXIT args
    0422'CF     00'  D0   061E  1279          MOVL    S^#SS$_BADPARAM,STATUS   ; Set the exit status
              06F8      31   0623  1280          BRW     ERROR_EXIT               ; Bitch and quit
```

```
                                                     0626  1282 ;+
                                                     0626  1283 ;       If the prompt flag is set prompt the user for the number of passes.  If
                                                     0626  1284 ;       it is not set, try to use P2 for the pass count.  If the prompt returns
                                                     0626  1285 ;       null, or if not prompting and P2 is null, we use the default, one pass.
                                                     0626  1286 ;-
                                                     0626  1287
                                                     0626  1288 PASS:
          1E 043A'CF    01    E1   0626  1289              BBC     #PROMPTV,FLAGS,3$        ; BR if not prompting
                  098B'CF       DF   062C  1290              PUSHAL  OUTLEN                   ; Set response length location
                  0366'CF       DF   0630  1291              PUSHAL  PASS_PROMPT              ; Set prompt string
                  0857'CF       DF   0634  1292              PUSHAL  ANSWER                   ; Set answer address
       00000000'GF    03       FB   0638  1293              CALLS   #3,G^LIB$GET_COMMAND     ; Ask for the pass count
                 19    50       E8   063F  1294              BLBS    R0,5$                    ; If no failure than continue
              0422'CF    50       D0   0642  1295              MOVL    R0,STATUS                ; else save error and
                    070D       31   0647  1296              BRW     FINI                     ; bail out
                                    064A  1297 3$:
       098B'CF    0443'CF       D0   064A  1298              MOVL    P2_DESC,OUTLEN           ; Set P2 param length in buffer
098F'DF 055A'CF    0443'CF       28   0651  1299              MOVC3   P2_DESC,P2_BUF,@OUTLEN+4 ; Put in defined pass count
                                    065B  1300 5$:
                  098B'CF       D5   065B  1301              TSTL    OUTLEN                   ; Do we have a value yet?
                       11       12   065F  1302              BNEQ    10$                      ; Br if yes...
              0997'CF    01       D0   0661  1303              MOVL    #1,PASS_COUNT            ; ...else save the integer default...
              098B'CF    01       D0   0666  1304              MOVL    #1,OUTLEN                ; ...and fill in the default pass count
              085F'CF    31       90   066B  1305              MOVB    #^A/1/,ANSWER+8
                    5E       11   0670  1306              BRB     20$                      ; Go to logical name create
                                    0672  1307
                                    0672  1308 10$:     ; Here we test for valid input - either from P2 or response to prompt
                                    0672  1309
                       04    DD   0672  1310              PUSHL   #4                       ; Push size of results
                  0997'CF    DF   0674  1311              PUSHAL  PASS_COUNT               ; Push place for results
                  0988'CF    DF   0678  1312              PUSHAL  OUTLEN                   ; Push ascii results
       00000000'GF    03    FB   067C  1313              CALLS   #3,G^OTS$CVT_TI_L        ; Save the long word pass count
       00000000'8F    50    D1   0683  1314              CMPL    R0,#OTS$_INPCONERR       ; Did it get input right?
                 44    12   068A  1315              BNEQ    20$                      ; Br if yes...
          05 043A'CF    01    E1   068C  1316              BBC     #PROMPTV,FLAGS,15$       ; BR if not prompting
                  063E    30   0692  1317              BSBW    SYNTAX_ERROR             ; ...else report the error...
                    8F    11   0695  1318              BRB     PASS                     ; ...and try again
                                    0697  1319
                                    0697  1320 15$:     ; P2 is an invalid string for pass count - bitch and quit
                                    0697  1321
                                    0697  1322              $FAO_S  CTRSTR = INVALID_PASS_MSG,-
                                    0697  1323                      OUTLEN = BUFFER_PTR,-
                                    0697  1324                      OUTBUF = FAO_BUF,-
                                    0697  1325                      P1     = #P2_DESC
              00000000'8F    DD   06B0  1326              PUSHL   #SS$_BADPARAM
                  0045'CF    DF   06B6  1327              PUSHAL  BUFFER_PTR
                       01    DD   06BA  1328              PUSHL   #1
              00741132    8F    DD   06BC  1329              PUSHL   #UETP$_TEXT!STS$K_ERROR
                       04    DD   06C2  1330              PUSHL   #4
0422'CF       00000000'8F    D0   06C4  1331              MOVL    #SS$_BADPARAM,STATUS     ; Set the exit status
                  064E    31   06CD  1332              BRW     ERROR_EXIT
                                    06D0  1333 20$:
                                    06D0  1334              $CRELOG_S LOGNAM = PASS_NAME,-
                                    06D0  1335                        EQLNAM = OUTLEN,-
                                    06D0  1336                        TBLFLG = #1            ; Make the pass count group logical name
```

```
                                06E3  1338 ;+
                                06E3  1339 ;       The default LOADS value is determined by several system parameters.
                                06E3  1340 ;       These parameters are extracted from the system and crunched to a
                                06E3  1341 ;       final value.  The system parameters are:
                                06E3  1342 ;
                                06E3  1343 ;               SID             CPU type, modified if multiprocessor config
                                06E3  1344 ;               MEM_FREE        Free main memory
                                06E3  1345 ;               MEM_MODIFY      Modified main memory
                                06E3  1346 ;               WS_SIZE         Current process working set size
                                06E3  1347 ;               FREE_PAGE       Free page file space
                                06E3  1348 ;               SWAP_SIZE       Free process swap slots
                                06E3  1349 ;
                                06E3  1350 ;       Constants are defined in this program for the calculation:
                                06E3  1351 ;
                                06E3  1352 ;               PP_PAGE_USAGE   Estimated amount of page file used per process
                                06E3  1353 ;               PER_WS_INUSE    Estimated amount of WS in constant use
                                06E3  1354 ;               CPU_SCALE       Estimated CPU performance ratio where 11/780 = 1
                                06E3  1355 ;
                                06E3  1356 ;       The equation used with these values is given in the strings DUMP_MSG1
                                06E3  1357 ;       and DUMP_MSG2.
                                06E3  1358 ;-
                                06E3  1359
                                06E3  1360 LOAD:
 57   09E7'CF   09EB'CF   C1    06E3  1361          ADDL3   MEM_MODIFY,MEM_FREE,R7  ; Calculate total amount of free memory
           57     57     4E    06EB  1362          CVTLF   R7,R7                    ; Convert free memory size to float
      58   09AF'CF     4E    06EE  1363          CVTLF   WS_SIZE,R8               ; Convert WS to floating format
 58    CCCD3F4C 8F      44    06F3  1364          MULF2   #PER_WS_INUSE,R8         ; Scale the WS
           57     58     46    06FA  1365          DIVF2   R8,R7                    ; Create a rough process capacity count
      57   0993'CF     44    06FD  1366          MULF2   CPU_SCALE,R7             ; Scale the count for the CPU type
           57     57     4A    0702  1367          CVTFL   R7,R7                    ; Convert back to integer
       000003E8 8F      C7    0705  1368          DIVL3   #PP_PAGE_USAGE,-         ; Calculate page process count limit
      56   09F3'CF             070B  1369                  PAGE_SIZE,R6
                                070F  1370
             02     DD          070F  1371          PUSHL   #2                       ; Push # of digits in the fraction
        02DE'CF     DF          0711  1372          PUSHAL  CPU_SCALE_DES            ; Push string storage desc adr
        0993'CF     DF          0715  1373          PUSHAF  CPU_SCALE                ; Push adr of floating number
00000000'GF     03  FB          0719  1374          CALLS   #3,G^FOR$CNV_OUT_F       ; Make the number a string
             02     DD          0720  1375          PUSHL   #2                       ; Push # of digits in the fraction
        02EA'CF     DF          0722  1376          PUSHAL  WS_INUSE_DES             ; Push string storage desc adr
        02F6'CF     DF          0726  1377          PUSHAF  WS_INUSE                 ; Push adr of floating number
00000000'GF     03  FB          072A  1378          CALLS   #3,G^FOR$CNV_OUT_F       ; Make the number a string
 76 043A'CF     04  E1          0731  1379          BBC     #DUMPV,FLAGS,10$         ; BR if not in dump mode - no message
                                0737  1380          $FAO_S  CTRSTR = DUMP_MSG1,-     ; Make the first output string
                                0737  1381                  OUTLEN = BUFFER_PTR,-
                                0737  1382                  OUTBUF = FAO_BUF,-
                                0737  1383                  P1     = #CPU_SCALE_DES,-
                                0737  1384                  P2     = MEM_FREE,-
                                0737  1385                  P3     = MEM_MODIFY,-
                                0737  1386                  P4     = WS_SIZE,-
                                0737  1387                  P5     = #WS_INUSE_DES,-
                                0737  1388                  P6     = R7
                                0764  1389          $PUTMSG_S MSGVEC = DUMP_MSG_PTR,- ; Print the filled in equation
                                0764  1390                    ACTRTN = ACTRTN
                                0777  1391          $FAO_S  CTRSTR = DUMP_MSG2,-     ; Make the second output string
                                0777  1392                  OUTLEN = BUFFER_PTR,-
                                0777  1393                  OUTBUF = FAO_BUF,-
                                0777  1394                  P1     = SWAP_SIZE,-
```

```
                     0777  1395                P2      = PAGE_SIZE,-
                     0777  1396                P3      = #PP_PAGE_USAGE-
                     0777  1397                P4      = R6
                     079A  1398        $PUTMSG_S MSGVEC = DUMP_MSG_PTR,- ; Print the filled in equation
                     079A  1399                  ACTRTN = ACTRTN
                     07AD  1400 10$:
    09F3'CF    56    D0   07AD  1401        MOVL    R6,PAGE_SIZE            ; Page process count limit
    09E3'CF    57    D0   07B2  1402        MOVL    R7,MEM_SIZE             ; Available main memory
        09EF'CF    DF   07B7  1403        PUSHAL  SWAP_SIZE               ; Find the minimum of swap slots...
        09F3'CF    DF   07BB  1404        PUSHAL  PAGE_SIZE               ; ...free page file space...
        09E3'CF    DF   07BF  1405        PUSHAL  MEM_SIZE                ; ...usable main memory...
00000000'GF    03    FB   07C3  1406        CALLS   #3,G^MTH$JMINO          ; ...and leave the results in R0
    099B'CF    50    D0   07CA  1407        MOVL    R0,LOAD_COUNT           ; save the MIN
    098B'CF    04    D0   07CF  1408        MOVL    #4,OUTLEN               ; Set the results length
        098B'CF    7F   07D4  1409        PUSHAQ  OUTLEN                  ; Push output string desc
        099B'CF    DF   07D8  1410        PUSHAL  LOAD_COUNT              ; Push the load count value
00000000'GF    02    FB   07DC  1411        CALLS   #2,G^OTS$CVT_L_TI       ; Convert the load count to a string
                     07E3  1412
                     07E3  1413        $SETSFM_S ENBFLG = #0              ; Disable SS failure mode if no match
                     07EC  1414        $DELLOG_S LOGNAM = USERS,-        ; Clean out any possible name that...
                     07EC  1415                  TBLFLG = #1             ; ...might be left from a previous run
                     07FB  1416        $SETSFM_S ENBFLG = #1             ; Re-enable system service failure mode
                     0804  1417
 46 043A'CF    01    E1   0804  1418        BBC     #PROMPTV,FLAGS,40$      ; BR if we need not prompt at all
        0B88'CF    39   080A  1419        MATCHC  KEY_LOAD_DESC,-         ; We need only prompt...
        0B8C'DF         080E  1420                @KEY_LOAD_DESC+4,-      ; ...if the LOAD phase...
01AA'CF    01A2'CF      0811  1421                PARAM_MSG,PARAM_BUF     ; ...was among the phases selected
              03    13   0817  1422        BEQL    20$                     ; BR if user has a choice
            00BC    31   0819  1423        BRW     80$                     ; No choice - use default
                     081C  1424 20$:    $FAO_S  CTRSTR = LOAD_PROMPT,-
                     081C  1425                 OUTLEN = BUFFER_PTR,-
                     081C  1426                 OUTBUF = FAO_BUF,-
                     081C  1427                 P1     = LOAD_COUNT     ; Create the prompt string
                     0833  1428
044B'CF    00FF  8F    B0   0833  1429 30$:    MOVW    #MAXSYM_SZ,P3_DESC      ; Define desc for response
        044B'CF    3F   083A  1430        PUSHAW  P3_DESC                 ; Set response length location
        0045'CF    DF   083E  1431        PUSHAL  BUFFER_PTR              ; Set prompt string
        044B'CF    7F   0842  1432        PUSHAQ  P3_DESC                 ; Set answer address
00000000'GF    03    FB   0846  1433        CALLS   #3,G^LIB$GET_COMMAND    ; Ask for the load count
              32    50   E9   084D  1434        BLBC    R0,50$                  ; BR if failure
                     0850  1435
                     0850  1436 40$:    ; Test for valid input from P3 or prompt response
    52    044B'CF    7E   0850  1437        MOVAQ   P3_DESC,R2              ; Point to desc for response
              62    B5   0855  1438        TSTW    (R2)                    ; Any response?
              7F    13   0857  1439        BEQL    80$                     ; BR if not - use default
        098B'CF    62    3C   0859  1440        MOVZWL  (R2),OUTLEN             ; Set P3 param length in buffer
098F'DF    04    B2    62    28   085E  1441        MOVC3   (R2),@4(R2),@OUTLEN+4   ; Use P3 for load count
        0436'CF    DF   0865  1442        PUSHAL  ARG_COUNT               ; Push place for results
        098B'CF    DF   0869  1443        PUSHAL  OUTLEN                  ; Push ascii results
00000000'GF    02    FB   086D  1444        CALLS   #2,G^OTS$CVT_TI_L       ; Save the long word load count
            5A    50   E8   0874  1445        BLBS    R0,70$                  ; BR if we got a reasonable number
 05 043A'CF    01    E1   0877  1446        BBC     #PROMPTV,FLAGS,50$      ; BR if not prompting
              0453    30   087D  1447        BSBW    SYNTAX_ERROR            ; ...else report the error...
              B1    11   0880  1448        BRB     30$                     ; ...and try again
                     0882  1449
    52    04    D0   0882  1450 50$:    MOVL    #4,R2                   ; Assume we are prompting
 11 043A'CF    01    E0   0885  1451        BBS     #PROMPTV,FLAGS,60$      ; BR if that's the case
```

```
                    00'  DD  088B  1452          PUSHL   S^#SS$_BADPARAM          ; We give an additional error if not
                    52   D6  088D  1453          INCL    R2
                    10   EF  088F  1454          EXTZV   #STS$V_FAC_NO,-          ; Was this a System or RMS error?
          53   50   0C       0891  1455                  #STS$S_FAC_NO,R0,R3
                    53   D7  0894  1456          DECL    R3                       ; They're facilities 0 & 1, respectively
                    04   15  0896  1457          BLEQ    60$                      ; BR if System or RMS
                    00   DD  0898  1458          PUSHL   #0                       ; Dummy arg count needed...
                    52   D6  089A  1459          INCL    R2                       ; ...for other facilities' messages
                             089C  1460 60$:
                    50   DD  089C  1461          PUSHL   R0                       ; Save the error status
                             089E  1462          $FAO_S  CTRSTR = INVALID_LOADCNT_MSG,-  ; P3 is an invalid load count
                             089E  1463                  OUTLEN = BUFFER_PTR,-
                             089E  1464                  OUTBUF = FAO_BUF,-
                             089E  1465                  P1     = #P3_DESC
              0045'CF    DF  08B7  1466          PUSHAL  BUFFER_PTR
                    01   DD  08BB  1467          PUSHL   #1
          00741132  8F   DD  08BD  1468          PUSHL   #UETP$_TEXT!STS$K_ERROR
                    52   DD  08C3  1469          PUSHL   R2
    0422'CF   00000000'8F   D0  08C5  1470          MOVL    #SS$_BADPARAM,STATUS     ; Set the exit status
                   044D   31  08CE  1471          BRW     ERROR_EXIT               ; Bitch and quit
                             08D1  1472
    099B'CF   0436'CF      D0  08D1  1473 70$:     MOVL    ARG_COUNT,LOAD_COUNT     ; It converted OK save it away
                             08D8  1474 80$:     $CRELOG_S LOGNAM = USERS,-
                             08D8  1475                    EQLNAM = OUTLEN,-
                             08D8  1476                    TBLFLG = #1              ; Make the load count group logical name
```

```
                                                08EB  1478 ;+
                                                08EB  1479 ; If the prompt flag is set we prompt the user for LONG or SHORT report format
                                                08EB  1480 ; to be used by the rest of the UETP, else if P4 is defined we use that.
                                                08EB  1481 ; If P4 is not defined or the prompt returns null, we use the default which
                                                08EB  1482 ; is LONG report.
                                                08EB  1483 ;-
                                                08EB  1484
                                                08EB  1485 REPORT_Q:
          13 043A'CF    01    E0                08EB  1486         BBS     #PROMPTV,FLAGS,3$       ; BR if prompting
     098B'CF    0453'CF    D0                    08F1  1487         MOVL    P4_DESC,OUTLEN          ; Set P4 param length in buffer
098F'DF   0758'CF    0453'CF    28              08F8  1488         MOVC3   P4_DESC,P4_BUF,@OUTLEN+4 ; Put specified mode in buffer
                        1E    11                0902  1489         BRB     5$
                                                0904  1490 3$:
                098B'CF    DF                   0904  1491         PUSHAL  OUTLEN                  ; Set response length location
                03D7'CF    DF                   0908  1492         PUSHAL  REPORT_PROMPT           ; Set prompt string
                0857'CF    DF                   090C  1493         PUSHAL  ANSWER                  ; Set asnwer address
          00000000'GF    03    FB                0910  1494         CALLS   #3,G^LIB$GET_COMMAND    ; Ask for the report format
                   08    50    E8                0917  1495         BLBS    R0,5$                   ; If no failure than continue
              0422'CF    50    D0                091A  1496         MOVL    R0,STATUS               ; else save error and
                      0435    31                091F  1497         BRW     FINI                    ; bail out
                                                0922  1498 5$:
                098B'CF    D5                   0922  1499         TSTL    OUTLEN                  ; Any format specified?
                      02    12                0926  1500         BNEQ    10$                     ; Br if yes...
                      59    11                0928  1501         BRB     20$                     ; Go fill in LONG
                                                092A  1502 10$:
             085F'CF    20    8A                092A  1503         BICB2   #LCBIT,ANSWER+8          ; Make sure that it is upper case
          085F'CF    4C    8F                  092F  1504         CMPB    #^A/L/,ANSWER+8         ; Is it long report format?
                      4C    13                0935  1505         BEQL    20$                     ; Br if yes
          085F'CF    53    8F                  0937  1506         CMPB    #^A/S/,ANSWER+8         ; Is it short report format?
                      54    13                093D  1507         BEQL    30$                     ; Br if yes...
          05 043A'CF    01    E1                093F  1508         BBC     #PROMPTV,FLAGS,15$      ; BR if not prompting
                      038B    30                0945  1509         BSBW    SYNTAX_ERROR            ; ...else report a syntax error...
                        A1    11                0948  1510         BRB     REPORT_Q                ; ...and ask again
                                                094A  1511
                                                094A  1512 15$:    ; P4 is an invalid report type - bitch and quit
                                                094A  1513
                                                094A  1514         $FAO_S  CTRSTR = INVALID_REPORT_MSG,-
                                                094A  1515                 OUTLEN = BUFFER_PTR,-
                                                094A  1516                 OUTBUF = FAO_BUF,-
                                                094A  1517                 P1     = #P4_DESC
                   00000000'8F    DD            0963  1518         PUSHL   #SS$_BADPARAM
                        0045'CF    DF            0969  1519         PUSHAL  BUFFER_PTR
                             01    DD            096D  1520         PUSHL   #1
                   00741132 8F    DD            096F  1521         PUSHL   #UETP$_TEXT!STS$K_ERROR
                             04    DD            0975  1522         PUSHL   #4
          0422'CF    00000000'8F    D0          0977  1523         MOVL    #SS$_BADPARAM,STATUS    ; Set the exit status
                      039B    31                0980  1524         BRW     ERROR_EXIT
                                                0983  1525 20$:    ; Long format
                0988'CF    04    B0             0983  1526         MOVW    #4,OUTLEN
          085F'CF    474E4F4C 8F    D0          0988  1527         MOVL    #^A/LONG/,ANSWER+8
                      12    11                  0991  1528         BRB     40$
                                                0993  1529 30$:    ; Short format
                0988'CF    05    B0             0993  1530         MOVW    #5,OUTLEN
085F'CF    00000054 524F4853 8F    7D          0998  1531         MOVQ    #^A/SHORT/,ANSWER+8
                                                09A5  1532 40$:
                                                09A5  1533         $CMKRNL_S LOGNAM = REPORT_NAME,-
                                                09A5  1534                   EQLNAM = OUTLEN,-
```

```
09A5  1535                      TBLFLG = #1             ; Make the report format group logical name
09B8  1536
09B8  1537 ;+
09B8  1538 ;
09B8  1539 ; Any additional UETP prompting code should be inserted at this point
09B8  1540 ; in the code.
09B8  1541 ;
09B8  1542 ;-
```

```
                        09B8  1544 ;+
                        09B8  1545 ;       All the interaction needed to set up a run of the UETP has been done.
                        09B8  1546 ;       Clean up and form a message summarizing what the user wants.
                        09B8  1547 ;-
                        09B8  1548 FINAL_MESSAGE:
                        09B8  1549          $FAO_S  CTRSTR = START_MESSAGE,-; Make the startup message
                        09B8  1550                  OUTLEN = BUFFER_PTR,-
                        09B8  1551                  OUTBUF = FAO_BUF,-
                        09B8  1552                  P1     = #0
09AB'CF   0045'CF   DE  09CD  1553          MOVAL   BUFFER_PTR,MSG_DESC
                        09D4  1554          $PUTMSG_S-
                        09D4  1555                  MSGVEC = VECTOR,-       ; Go ahead and output msg
                        09D4  1556                  ACTRTN = ACTRTN         ; Output it to log file as well
     56   01A2'CF   3C  09E7  1557          MOVZWL  PARAM_MSG,R6            ; Get current length
0442'DF   043E'CF   28  09EC  1558          MOVC3   PHASES,@PHASES+4,-      ; Add "PHASE(S)" to  msg
        01AA'CF46       09F3  1559                  PARAM_BUF[R6]
01A2'CF   043E'CF   A0  09F7  1560          ADDW2   PHASES,PARAM_MSG        ; Update msg length
                        09FE  1561
     0997'CF   01   D1  09FE  1562          CMPL    #1,PASS_COUNT           ; Are we running only 1 pass
                 05   12  0A03  1563          BNEQ    20$                     ; Br if not 1
     0179'CF   02   A2  0A05  1564          SUBW2   #2,PASS_MSG             ; Drop "ES" off "PASSES"
                        0A0A  1565 20$:
                        0A0A  1566          $FAO_S  CTRSTR = PASS_MSG,-     ; Create pass count portion of start msg
                        0A0A  1567                  OUTLEN = BUFFER_PTR,-
                        0A0A  1568                  OUTBUF = FAO_BUF,-
                        0A0A  1569                  P1     = PASS_COUNT
     56   01A2'CF   3C  0A21  1570          MOVZWL  PARAM_MSG,R6            ; Get current length
0049'DF   0045'CF   28  0A26  1571          MOVC3   BUFFER_PTR,@BUFFER_PTR+4,- ; Add number of pass(es) to  msg
        01AA'CF46       0A2D  1572                  PARAM_BUF[R6]
01A2'CF   0045'CF   A0  0A31  1573          ADDW2   BUFFER_PTR,PARAM_MSG    ; Update msg length
                        0A38  1574          $FAO_S  CTRSTR = LOAD_MSG,-     ; Create loads count part of start msg
                        0A38  1575                  OUTLEN = BUFFER_PTR,-
                        0A38  1576                  OUTBUF = FAO_BUF,-
                        0A38  1577                  P1     = LOAD_COUNT
     56   01A2'CF   3C  0A4F  1578          MOVZWL  PARAM_MSG,R6            ; Get current length
0049'DF   0045'CF   28  0A54  1579          MOVC3   BUFFER_PTR,@BUFFER_PTR+4,- ; Add number of load(s) to msg
        01AA'CF46       0A5B  1580                  PARAM_BUF[R6]
01A2'CF   0045'CF   A0  0A5F  1581          ADDW2   BUFFER_PTR,PARAM_MSG    ; Update msg length
                        0A66  1582 30$:
     56   01A2'CF   3C  0A66  1583          MOVZWL  PARAM_MSG,R6            ; Get current length
  085F'CF    4C 8F   91  0A6B  1584          CMPB    #^A/L7,ANSWER+8         ; Long report?
                 14   12  0A71  1585          BNEQ    40$                     ; Br if not Long
0451'DF   044D'CF   28  0A73  1586          MOVC3   LONG_MSG,@LONG_MSG+4,-  ; Add "LONG REPORT" to msg
        01AA'CF46       0A7A  1587                  PARAM_BUF[R6]
01A2'CF   044D'CF   A0  0A7E  1588          ADDW2   LONG_MSG,PARAM_MSG      ; Update length
                 12   11  0A85  1589          BRB     50$                     ; Go output message
                        0A87  1590 40$:
0469'DF   0465'CF   28  0A87  1591          MOVC3   SHORT_MSG,@SHORT_MSG+4,- ; Add "SHORT REPORT" too msg
        01AA'CF46       0A8E  1592                  PARAM_BUF[R6]
01A2'CF   0465'CF   A0  0A92  1593          ADDW2   SHORT_MSG,PARAM_MSG     ; Update msg length
                        0A99  1594 50$:
09AB'CF   01A2'CF   DE  0A99  1595          MOVAL   PARAM_MSG,MSG_DESC
                        0AA0  1596          $PUTMSG_S-
                        0AA0  1597                  MSGVEC = VECTOR,-       ; Go ahead and output msg
                        0AA0  1598                  ACTRTN = ACTRTN         ; Output it to log file as well
0422'CF   10000000'8F   D0  0AB3  1599          MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                        0ABC  1600          $EXIT_S STATUS                  ; Exit with the status
```

```
                                0AC7  1602              .SBTTL  Figure Various Limits of This Configuration
                                0AC7  1603 ;++
                                0AC7  1604 ; This code was stolen from the CLI Utility program for SHOW MEMORY.  It runs
                                0AC7  1605 ; in EXEC mode.
                                0AC7  1606 ;
                                0AC7  1607 ; It uses the memory descriptors in the Restart Parameter Block
                                0AC7  1608 ; to determine the amount of physical memory in use.  A check is made to
                                0AC7  1609 ; see if multiport memory should be counted as local memory.
                                0AC7  1610 ;
                                0AC7  1611 ; The following set of assumptions state that all multiport memory adapter
                                0AC7  1612 ; type codes are bounded by NDT$_MPM0 and NDT$_MPM3 and that no adapter
                                0AC7  1613 ; type codes in this range represent anything other than multiport memory.
                                0AC7  1614 ;
                                0AC7  1615              ASSUME  NDT$_MPM0 LT NDT$_MPM1
                                0AC7  1616              ASSUME  NDT$_MPM1 LT NDT$_MPM2
                                0AC7  1617              ASSUME  NDT$_MPM2 LT NDT$_MPM3
                                0AC7  1618 ;--
                                0AC7  1619
                                0AC7  1620 GET_MEM_INFO:
                        00FC    0AC7  1621              .WORD   ^M<R2,R3,R4,R5,R6,R7>
                                0AC9  1622
    50   00000000'GF    D0    0AC9  1623              MOVL    G^EXE$GL_CONFREGL,R0    ; Get address of TR/adapter type array
    51   00000000'GF    D0    0AD0  1624              MOVL    G^EXE$GL_RPB,R1         ; Get addr of RPB
         52    008C C1  DE    0AD7  1625              MOVAL   RPB$L_MEMDSC(R1),R2     ; Get addr of memory descriptors
                  56    7C    0ADC  1626              CLRQ    R6                      ; Init local and shared page counts
                  62    D5    0ADE  1627 10$:         TSTL    (R2)                    ; End of memdsc list?
                  27    13    0AE0  1628              BEQL    40$                     ; Yes - finished collecting info
    53   62    08    18    EF    0AE2  1629              EXTZV   #RPB$V_TR,#RPB$S_TR,(R2),R3 ; Get TR number
              53    6043  D0    0AE7  1630              MOVL    (R0)[R3],R3             ; Convert to adapter type
                    00    EF    0AEB  1631              EXTZV   #RPB$V_PAGCNT,-         ; Get page count
         54   62    18          0AED  1632                      #RPB$S_PAGCNT,(R2),R4
           40 8F    53    91    0AF0  1633              CMPB    R3,#NDT$_MPM0           ; Is adapter number below MPM range?
                  0B    1F    0AF4  1634              BLSSU   20$                     ; If so, this is local memory
           43 8F    53    91    0AF6  1635              CMPB    R3,#NDT$_MPM3           ; Is adapter number above MPM range
                  05    1A    0AFA  1636              BGTRU   20$                     ; If so, this is also local memory
              57    54    C0    0AFC  1637              ADDL2   R4,R7                   ; Otherwise, this is multiport memory
                  03    11    0AFF  1638              BRB     30$                     ; Go to end of loop
                              0B01  1639
              56    54    C0    0B01  1640 20$:         ADDL2   R4,R6                   ; This is local memory
              52    08    C0    0B04  1641 30$:         ADDL2   #RPB$C_MEMDSCSIZ,R2     ; Point to next memory descriptor
                  D5    11    0B07  1642              BRB     10$                     ;  and go back to top of loop
                              0B09  1643
                              0B09  1644 ; There are four cases that can occur here.
                              0B09  1645 ;
                              0B09  1646 ; 1.  There are no multiport memory controllers on the system.  R7 is 0 already.
                              0B09  1647 ;
                              0B09  1648 ; 2.  Multiport memory is being used as global shared memory.  Must clear R7.
                              0B09  1649 ;
                              0B09  1650 ; 3.  Multiport memory is being used as local memory.  This case is
                              0B09  1651 ;     distinguished by RPB$V_USEMPM being set in the RPB copy of R5.
                              0B09  1652 ;
                              0B09  1653 ; 4.  Only multiport memory is being used as local memory.  Any memory
                              0B09  1654 ;     in local controllers is ignored.  This is the multiprocessor
                              0B09  1655 ;     configuration.  This case is distinguished by RPB$V_MPM
                              0B09  1656 ;     being set in the RPB copy of R5.  Must clear R6.
                              0B09  1657
                        0B    E1    0B09  1658 40$:      BBC     #RPB$V_MPM,-            ; BR if not multiprocessor config
```

```
                04 30 A1        0B0B  1659                    RPB$L_BOOTR5(R1),50$
                      56  D4    0B0E  1660            CLRL    R6                      ; We don't count any local memory...
                      07  11    0B10  1661            BRB     60$                     ; ...but always count shared memory
                      0C  E0    0B12  1662  50$:       BBS     #RPB$V_USEMPM,-         ; Also count shared memory?
                02 30 A1        0B14  1663                    RPB$L_BOOTR5(R1),60$
                      57  D4    0B17  1664            CLRL    R7                      ; No
                   57   56  C0  0B19  1665  60$:       ADDL2   R6,R7                   ; Calculate total available memory
           57   00000000'GF  D1  0B1C  1666            CMPL    G^MMG$GL_PHYPGCNT,R7    ; How does that compare with SYSGEN?
                      07  1E    0B23  1667            BGEQU   70$                     ; BR if we may use all we have
           57   00000000'GF  D0  0B25  1668            MOVL    G^MMG$GL_PHYPGCNT,R7    ; Use only as much as we are allowed
        09E3'CF    57  D0        0B2C  1669  70$:       MOVL    R7,MEM_SIZE             ; Return what's around and allowed
                                 0B31  1670
09E7'CF 00000000'GF  D0          0B31  1671            MOVL    G^SCH$GL_FREECNT,MEM_FREE ; Get number of free memory pages
09EB'CF 00000000'GF  D0          0B3A  1672            MOVL    G^SCH$GL_MFYCNT,MEM_MODIFY ; Get number of memory pages modified
        00000000'GF  A3          0B43  1673            SUBW3   G^SCH$GW_PROCCNT,-      ; Process slots free = total slots...
     50 00000000'GF              0B49  1674                    G^SCH$GW_PROCLIM,R0     ; ...less slots in use...
           50    02  A2          0B4F  1675            SUBW2   #2,R0                   ; ...less slots for swapper and null...
        09EF'CF  50  3C          0B52  1676            MOVZWL  R0,SWAP_SIZE            ; ...converted to a form we can use
              50   00'  D0        0B57  1677            MOVL    S^#SS$_NORMAL,R0
                        04      0B5A  1678            RET
```

```
                                D 5
0B5B  1680          .SBTTL  System Service Exception Handler
0B5B  1681 ;++
0B5B  1682 ; FUNCTIONAL DESCRIPTION:
0B5B  1683 ;       This routine is executed if a system service or RMS error occurs or
0B5B  1684 ;       if a LIB$SIGNAL system service is used to output a message.
0B5B  1685 ;       Information about this method of handling messages and errors can be
0B5B  1686 ;       found in the VMS COMMON RUN-TIME manual and in the VMS SYSTEM SERVICE
0B5B  1687 ;       manual.
0B5B  1688 ;
0B5B  1689 ; CALLING SEQUENCE:
0B5B  1690 ;       Entered via an exception from the system
0B5B  1691 ;
0B5B  1692 ; INPUT PARAMETERS:
0B5B  1693 ;       ERROR_COUNT    = previous cumulative error count
0B5B  1694 ;
0B5B  1695 ;          AP ---->     +---------------+
0B5B  1696 ;                       |       2       |
0B5B  1697 ;                       +---------------+
0B5B  1698 ;                       | SIGNL ARY PNT |
0B5B  1699 ;                       +---------------+
0B5B  1700 ;                       | MECH  ARY PNT |  ---------
0B5B  1701 ;                       +---------------+          ^
0B5B  1702 ;                       |       4       |          |
0B5B  1703 ;                       +---------------+          |
0B5B  1704 ;                       | ESTABLISH FP  |          |
0B5B  1705 ;                       +---------------+          |
0B5B  1706 ;                       |     DEPTH     |  Mechanism Array
0B5B  1707 ;                       +---------------+          |
0B5B  1708 ;                       |       R0      |          |
0B5B  1709 ;                       +---------------+          |
0B5B  1710 ;                       |       R1      |          v
0B5B  1711 ;                       +---------------+  ---------
0B5B  1712 ;                       |       N       |          ^
0B5B  1713 ;                       +---------------+          |
0B5B  1714 ;                       | CONDITION NAME|          |
0B5B  1715 ;                       +---------------+          |
0B5B  1716 ;                       | N-3 ADDITIONAL|    Signal Array
0B5B  1717 ;                       | LONG WORD ARGS|          |
0B5B  1718 ;                       +---------------+          |
0B5B  1719 ;                       |       PC      |          |
0B5B  1720 ;                       +---------------+          |
0B5B  1721 ;                       |       PSL     |          v
0B5B  1722 ;                       +---------------+  ---------
0B5B  1723 ; IMPLICIT INPUTS:
0B5B  1724 ;       NONE
0B5B  1725 ;
0B5B  1726 ; OUTPUT PARAMETERS:
0B5B  1727 ;       NONE
0B5B  1728 ;
0B5B  1729 ; IMPLICIT OUTPUTS:
0B5B  1730 ;       The messages are output to SYS$OUTPUT and to UETP.LOG.
0B5B  1731 ;
0B5B  1732 ; COMPLETION CODES:
0B5B  1733 ;       NONE
0B5B  1734 ;
0B5B  1735 ; SIDE EFFECTS:
0B5B  1736 ;       NONE
0B5B  1737 ;--
```

```
                        0B5B  1737
                        0B5B  1738 SSERROR:
                   0FFC 0B5B  1739          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0B5D  1740
                        0B5D  1741          $SETAST_S ENBFLG = #0                       ; Disable AST delivery
                01   DD 0B66  1742          PUSHL   #1                                  ; Assume ASTs were enabled
              50 00'    D1 0B68  1743          CMPL    S^#SS$_WASSET,R0                    ; Were ASTs enabled?
                   02   13 0B6B  1744          BEQL    10$                                 ; BR if they were
                   6E   D4 0B6D  1745          CLRL    (SP)                                ; Set ASTs to remain disabled
                        0B6F  1746 10$:
                        0B6F  1747          $SETSFM_S ENBFLG = #0                       ; Disable SS failure mode
                01   DD 0B78  1748          PUSHL   #1                                  ; Assume SS failure mode was enabled
              50 00'    D1 0B7A  1749          CMPL    S^#SS$_WASSET,R0                    ; Was SS failure mode enabled?
                   02   13 0B7D  1750          BEQL    20$                                 ; BR if it was
                   6E   D4 0B7F  1751          CLRL    (SP)                                ; Set SS failure mode to remain off
                        0B81  1752 20$:
           56   04 AC    D0 0B81  1753          MOVL    CHF$L_SIGARGLST(AP),R6      ; Get the signal array pointer
           59   04 A6    7D 0B85  1754          MOVQ    CHF$L_SIG_NAME(R6),R9       ; Get NAME in R9 and ARG1 in R10
                10   ED 0B89  1755          CMPZV   #STS$V_FAC_NO,-             ; Is this a message from LIB$SIGNAL?
                   0C      0B8B  1756                  #STS$S_FAC_NO,-
00000074 8F        59      0B8C  1757                  R9,#UETP$_FACILITY
                16   12 0B92  1758          BNEQ    30$                         ; BR if this is not a UETP exception
              66   02   C2 0B94  1759          SUBL2   #2,CHF$L_SIG_ARGS(R6)       ; Drop the PC and PSL
                        0B97  1760          $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6),- ; Print the message
                        0B97  1761                    ACTRTN = ACTRTN
                21   11 0BA8  1762          BRB     40$                         ; Restore ASTs and SS fail mode
                        0BAA  1763 30$:
59 00000000'8F    D1 0BAA  1764          CMPL    #SS$_SSFAIL,R9              ; RMS failures are SysSvc failures
                32   12 0BB1  1765          BNEQ    50$                         ; BR if this can't be an RMS failure
                10   ED 0BB3  1766          CMPZV   #STS$V_FAC_NO,-             ; Is it an RMS failure?
                   0C      0BB5  1767                  #STS$S_FAC_NO,-
              01   5A      0BB6  1768                  R10,#RMS$_FACILITY
                2B   12 0BBB  1769          BNEQ    50$                         ; BR if not
5A F0000000    8F   CA 0BBA  1770          BICL2   #^XF0000000,R10             ; Strip control bits from status code
     08 A6    04   39 0BC1  1771          MATCHC  #4,CHF$L_SIG_ARG1(R6),-     ; Is it an RMS failure for which...
                14      0BC5  1772                  #NRAT_LENGTH,-
           00A4'CF      0BC6  1773                  NO_RMS_AST_TABLE            ; ...no AST can be delivered?
                1A   13 0BC9  1774          BEQL    50$                         ; BR if so - must give error here
                        0BCB  1775 40$:
                01   BA 0BCB  1776          POPR    #^M<R0>                     ; Restore SS failure mode...
                        0BCD  1777          $SETSFM_S ENBFLG = R0               ; ...
                01   BA 0BD6  1778          POPR    #^M<R0>                     ; Restore AST enable...
                        0BD8  1779          $SETAST_S ENBFLG = R0               ; ...
              50 00'    D0 0BE1  1780          MOVL    S^#SS$_NORMAL,R0            ; Supply a standard status for exit
                   04   04 0BE4  1781          RET                                 ; Resume processing (or goto RMS_ERROR)
                        0BE5  1782 50$:
        0422'CF    59   D0 0BE5  1783          MOVL    R9,STATUS                   ; Save the status
                   58   D4 0BEA  1784          CLRL    R8                          ; Assume for now it's not SS failure
59 00000000'8F    D1 0BEC  1785          CMPL    #SS$_SSFAIL,R9              ; But is it a System Service failure?
                38   12 0BF3  1786          BNEQ    70$                         ; BR if not - no special case message
                        0BF5  1787          $GETMSG_S MSGID = R10,-             ; Get SS failure code associated text
                        0BF5  1788                    MSGLEN = BUFFER_PTR,-
                        0BF5  1789                    BUFADR = FAO_BUF,-
                        0BF5  1790                    FLAGS  = #14,-
                        0BF5  1791                    OUTADR = MSG_BLOCK
           0406'CF      95   0C0C  1792          TSTB    MSG_BLOCK+1                 ; Get FAO arg count for SS failure code
                16   13 0C10  1793          BEQL    60$                         ; Don't use $GETMSG if no $FAO args...
```

```
                0045'CF    DF  0C12  1794                PUSHAL  BUFFER_PTR                      ; ...else build up...
                     01    DD  0C16  1795                PUSHL   #1                              ; ...a message describing...
           00741130 8F     DD  0C18  1796                PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
                 00    5A  F0  0C1E  1797                INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
                 6E    03      0C21  1798                        #STS$S_SEVERITY,(SP)            ; ...the correct severity code
                 58    03  D0  0C23  1799                MOVL    #3,R8                           ; Count the number of args we pushed
                       05  11  0C26  1800                BRB     70$
                           0C28  1801 60$:
                       5A  DD  0C28  1802                PUSHL   R10                             ; Save SS failure code
                 58    01  D0  0C2A  1803                MOVL    #1,R8                           ; Count the number of args we pushed
                           0C2D  1804 70$:
              57 66    04  C5  0C2D  1805                MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
                 5E    57  C2  0C31  1806                SUBL2   R7,SP                           ; Save the current signal array...
           6E 04 A6    57  28  0C34  1807                MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
              7E 66    58  C1  0C39  1808                ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                    0ODE   31  0C3D  1809                BRW     ERROR_EXIT
                           0C40  1810
                           0C40  1811 ACTRTN:
                      0004 0C40  1812                    .WORD   ^M<R2>
                 52    04 AC  D0  0C42  1813                MOVL    4(AP),R2                        ; get the message descriptor address
              0A82'CF    62  3C  0C46  1814                MOVZWL  (R2),LOG_RAB+RAB$W_RSZ          ; set the message size
           0A88'CF 04    A2  D0  0C4B  1815                MOVL    4(R2),LOG_RAB+RAB$L_RBF         ; set the message address
                           0C51  1816                $PUT    RAB = LOG_RAB                   ; write to the log file
        50 00000000'8F     D0  0C5C  1817                MOVL    #SS$_NORMAL,R0                  ; set the return status code
                           04  0C63  1818                RET
                               0C64  1819
```

```
                                                G 5
                     0C64  1821                .SBTTL  RMS Error Handler
                     0C64  1822 ;++
                     0C64  1823 ; FUNCTIONAL DESCRIPTION:
                     0C64  1824 ;       This routine handles error returns from RMS calls.
                     0C64  1825 ;
                     0C64  1826 ; CALLING SEQUENCE:
                     0C64  1827 ;       Called by RMS when a file processing error is found.
                     0C64  1828 ;
                     0C64  1829 ; INPUT PARAMETERS:
                     0C64  1830 ;       The FAB or RAB associated with the RMS call.
                     0C64  1831 ;
                     0C64  1832 ; IMPLICIT INPUTS:
                     0C64  1833 ;       NONE
                     0C64  1834 ;
                     0C64  1835 ; OUTPUT PARAMETERS:
                     0C64  1836 ;       NONE
                     0C64  1837 ;
                     0C64  1838 ; IMPLICIT OUTPUTS:
                     0C64  1839 ;       Error message
                     0C64  1840 ;
                     0C64  1841 ; COMPLETION CODES:
                     0C64  1842 ;       NONE
                     0C64  1843 ;
                     0C64  1844 ; SIDE EFFECTS:
                     0C64  1845 ;       Program may exit, depending on severity of the error.
                     0C64  1846 ;
                     0C64  1847 ;--
                     0C64  1848
                     0C64  1849 RMS_ERROR:
                OFFC 0C64  1850         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0C66  1851
          56  04 AC   D0  0C66  1852         MOVL    4(AP),R6                ; See whether we're dealing with...
              66 03   91  0C6A  1853         CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                 16   12  0C6D  1854         BNEQ    10$                     ; BR if it's a RAB
          57 02C3'CF   DE  0C6F  1855         MOVAL   FILE,R7                 ; FAB-specific code:  text string...
             58  56   D0  0C74  1856         MOVL    R6,R8                   ; ...address of FAB...
                0C A6   DD  0C77  1857         PUSHL   FAB$L_STV(R6)           ; ...STV field for error...
                08 A6   DD  0C7A  1858         PUSHL   FAB$L_STS(R6)           ; ...STS field for error...
       0422'CF  08 A6   D0  0C7D  1859         MOVL    FAB$L_STS(R6),STATUS    ; ...and save the error code
                   15   11  0C83  1860         BRB     COMMON                  ; FAB and RAB share other code
                     0C85  1861 10$:
          57 02CF'CF   DE  0C85  1862         MOVAL   RECORD,R7               ; RAB-specific code:  text string...
             58  3C A6   D0  0C8A  1863         MOVL    RAB$L_FAB(R6),R8        ; ...address of associated FAB...
                0C A6   DD  0C8E  1864         PUSHL   RAB$L_STV(R6)           ; ...STV field for error...
                08 A6   DD  0C91  1865         PUSHL   RAB$L_STS(R6)           ; ...STS field for error...
       0422'CF  08 A6   D0  0C94  1866         MOVL    RAB$L_STS(R6),STATUS    ; ...and save the error code
                     0C9A  1867 COMMON:
             5A  34 A8   9A  0C9A  1868         MOVZBL  FAB$B_FNS(R8),R10       ; Get file name size for implicit PUSHL
                     0C9E  1869         $FAO_S  CTRSTR = RMS_ERR_STRING,-       ; Common code, prepare error message...
                     0C9E  1870                 OUTLEN = BUFFER_PTR,-
                     0C9E  1871                 OUTBUF = FAO_BUF,-
                     0C9E  1872                 P1     = R7,-
                     0C9E  1873                 P2     = R10,-
                     0C9E  1874                 P3     = FAB$L_FNA(R8)
          0045'CF    DF  0CB8  1875         PUSHAL  BUFFER_PTR              ; ...and arguments for ERROR_EXIT...
                01   DD  0CBC  1876         PUSHL   #1                      ; ...
       00741130 8F   DD  0CBE  1877         PUSHL   #UETP$_TEXT             ; ...
```

```
                00  EF  0CC4  1878          EXTZV   #STS$V_SEVERITY,-
                03      0CC6  1879                  #STS$S_SEVERITY,-
            59  0422'CF     0CC7  1880                  STATUS,R9               ; ...get the severity code...
                6E  59  88  0CCB  1881          BISB2   R9,(SP)                 ; ...and add it into the signal name
                    05  DD  0CCE  1882          PUSHL   #5                      ; Current arg count
                  004B  31  0CD0  1883          BRW     ERROR_EXIT
```

```
                     0CD3  1885          .SBTTL  Syntax Error Routine
                     0CD3  1886 ;++
                     0CD3  1887 ; FUNCTIONAL DESCRIPTION:
                     0CD3  1888 ;       This routine handles syntax errors.
                     0CD3  1889 ;
                     0CD3  1890 ; CALLING SEQUENCE:
                     0CD3  1891 ;       BSBW SYNTAX_ERROR
                     0CD3  1892 ;
                     0CD3  1893 ; INPUT PARAMETERS:
                     0CD3  1894 ;       NONE
                     0CD3  1895 ;
                     0CD3  1896 ; IMPLICIT INPUTS:
                     0CD3  1897 ;       NONE
                     0CD3  1898 ;
                     0CD3  1899 ; OUTPUT PARAMETERS:
                     0CD3  1900 ;       NONE
                     0CD3  1901 ;
                     0CD3  1902 ; IMPLICIT OUTPUTS:
                     0CD3  1903 ;       NONE
                     0CD3  1904 ;
                     0CD3  1905 ; COMPLETION CODES:
                     0CD3  1906 ;       NONE
                     0CD3  1907 ;
                     0CD3  1908 ; SIDE EFFECTS:
                     0CD3  1909 ;       NONE
                     0CD3  1910 ;
                     0CD3  1911 ;--
                     0CD3  1912
                     0CD3  1913 SYNTAX_ERROR:
                     0CD3  1914
      00D9'CF    DF  0CD3  1915          PUSHAL  SYNTAX_ERROR_MSG
            01    DD  0CD7  1916          PUSHL   #1
      00741132    8F  DD    0CD9  1917          PUSHL   #UETP$_TEXT!STS$K_ERROR
00000000'GF       03  FB    0CDF  1918          CALLS   #3,G^LIB$SIGNAL
                      05    0CE6  1919          RSB
```

```
                                                J 5
                     0CE7  1921          .SBTTL  CTRL/C Handler
                     0CE7  1922 ;++
                     0CE7  1923 ; FUNCTIONAL DESCRIPTION:
                     0CE7  1924 ;       This routine handles CTRL/C AST's
                     0CE7  1925 ;
                     0CE7  1926 ; CALLING SEQUENCE:
                     0CE7  1927 ;       Called via AST
                     0CE7  1928 ;
                     0CE7  1929 ; INPUT PARAMETERS:
                     0CE7  1930 ;       NONE
                     0CE7  1931 ;
                     0CE7  1932 ; IMPLICIT INPUTS:
                     0CE7  1933 ;       NONE
                     0CE7  1934 ;
                     0CE7  1935 ; OUTPUT PARAMETERS:
                     0CE7  1936 ;       NONE
                     0CE7  1937 ;
                     0CE7  1938 ; IMPLICIT OUTPUTS:
                     0CE7  1939 ;       NONE
                     0CE7  1940 ;
                     0CE7  1941 ; COMPLETION CODES:
                     0CE7  1942 ;       NONE
                     0CE7  1943 ;
                     0CE7  1944 ; SIDE EFFECTS:
                     0CE7  1945 ;       NONE
                     0CE7  1946 ;
                     0CE7  1947 ;--
                     0CE7  1948
                     0CE7  1949 CCASTHAND:
                0FFC 0CE7  1950          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0CE9  1951
          00B8'CF    DF   0CE9  1952          PUSHAL  CNTRLCMSG                       ; Set message pointer
                01   DD   0CED  1953          PUSHL   #1                              ; Set arg count
          00741130  8F   DD   0CEF  1954          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                00   DD   0CF5  1955          PUSHL   #0                              ; Indicate an abnormal termination
          000F'CF    DF   0CF7  1956          PUSHAL  TEST_NAME                       ; ...
                02   DD   0CFB  1957          PUSHL   #2                              ; ...
          007410E0  8F   DD   0CFD  1958          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
    00000000'GF    07   FB   0D03  1959          CALLS   #7,G^LIB$SIGNAL                 ; Output the message
                   D0   0D0A  1960          MOVL    #<STS$M_INHIB_MSG!-             ; Set the exit status
                        0D0B  1961                  SS$_CONTROLC-=
                        0D0B  1962                  STS$K_SUCCESS+STS$K_WARNING>,-
0422'CF    0FFFFFFF'8F      0D0B  1963                  STATUS
                        0D13  1964          $EXIT_S STATUS                          ; Terminate program cleanly
```

```
                                                .SBTTL  Error Exit
                                        ;++
                                        ; FUNCTIONAL DESCRIPTION:
                                        ;       This routine prints an error message and exits.
                                        ;
                                        ; CALLING SEQUENCE:
                                        ;       MOVx  error status value,STATUS
                                        ;       PUSHx error specific information on the stack
                                        ;       PUSHL current argument count
                                        ;       BRW   ERROR_EXIT
                                        ;
                                        ; INPUT PARAMETERS:
                                        ;       Arguments to LIB$SIGNAL, as above
                                        ;
                                        ; IMPLICIT INPUTS:
                                        ;       NONE
                                        ;
                                        ; OUTPUT PARAMETERS:
                                        ;       Message to SYS$OUTPUT and SYS$ERROR
                                        ;
                                        ; IMPLICIT OUTPUTS:
                                        ;       Program exit
                                        ;
                                        ; COMPLETION CODES:
                                        ;       NONE
                                        ;
                                        ; SIDE EFFECTS:
                                        ;       NONE
                                        ;
                                        ;--
                                        ERROR_EXIT:
                                        ;
0436'CF    08   8E   C1                 ADDL3   (SP)+,#8,ARG_COUNT      ; Get total # args, pop partial count
           0039'CF   D6                 INCL    ERROR_COUNT             ; Keep running error count
                00   DD                 PUSHL   #0                      ; Push the time parameter
           000F'CF   DF                 PUSHAL  TEST_NAME               ; Push test name...
      000F0002  8F   DD                 PUSHL   #^XF0002                ; ...arg count...
      007410E2  8F   DD                 PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
           0039'CF   DD                 PUSHL   ERROR_COUNT             ; Finish off arg list...
           000F'CF   DF                 PUSHAL  TEST_NAME               ; ...
      00010002  8F   DD                 PUSHL   #^X10002                ; ...
      00748022  8F   DD                 PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
00000000'GF   0436'CF   FB             CALLS   ARG_COUNT,G^LIB$SIGNAL  ; Truly bitch
                                        FINI:
           0422'CF   D5                 TSTL    STATUS                  ; Was an exit status supplied?
                09   12                 BNEQ    10$                     ; BR if one was
      007410E2  8F   D0                 MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; None there, supply a default
           0422'CF                              STATUS
                                        10$:
0422'CF 10000000  8F   C8               BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                                        $EXIT_S STATUS                  ; Exit in error
```

```
                        0D7A  2019          .SBTTL  Exit Handler
                        0D7A  2020 ;++
                        0D7A  2021 ; FUNCTIONAL DESCRIPTION:
                        0D7A  2022 ;       This routine handles cleanup on exits.
                        0D7A  2023 ;
                        0D7A  2024 ; CALLING SEQUENCE:
                        0D7A  2025 ;       Invoked automatically by $EXIT System Service.
                        0D7A  2026 ;
                        0D7A  2027 ; INPUT PARAMETERS:
                        0D7A  2028 ;       Location STATUS contains the exit status
                        0D7A  2029 ;
                        0D7A  2030 ; IMPLICIT INPUTS:
                        0D7A  2031 ;       NONE
                        0D7A  2032 ;
                        0D7A  2033 ; OUTPUT PARAMETERS:
                        0D7A  2034 ;       NONE
                        0D7A  2035 ;
                        0D7A  2036 ; IMPLICIT OUTPUTS:
                        0D7A  2037 ;       Various files are de-accessed, the process name is reset, and any
                        0D7A  2038 ;       necessary synchronization with UETPDEV01 is carried out.
                        0D7A  2039 ;
                        0D7A  2040 ; COMPLETION CODES:
                        0D7A  2041 ;       NONE
                        0D7A  2042 ;
                        0D7A  2043 ; SIDE EFFECTS:
                        0D7A  2044 ;       NONE
                        0D7A  2045 ;
                        0D7A  2046 ;--
                        0D7A  2047
                        0D7A  2048 EXIT_HANDLER:
                  0FFC  0D7A  2049          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0D7C  2050
                        0D7C  2051          $SETSFM_S ENBFLG = #0               ; Turn off System Service failure mode
                        0D85  2052          $CLOSE FAB = LOG_FAB                ; Close the log file
                        0D90  2053          $SETPRN_S PRCNAM = ACNT_NAME        ; Reset the process name
                     04 0D9B  2054          RET                                 ; That's all folks!
                        0D9C  2055
                        0D9C  2056          .END    UETINIT00
```

```
$$.TAB                  = 00000A60 R      03
$$.TABEND               = 00000AA4 R      03
$$.TMP                  = 00000000
$$.TMP1                 = 00000001
$$.TMP2                 = 000000CF
$$.TMPX                 = 00000000 R      04
$$.TMPX1                = 00000008
$$T1                    = 00000000
$$T2                    = 00000006
A730                      00000983 R      02
A750                      0000097C R      02
A780                      00000975 R      02
A782                      00000989 R      02
A785                      000009C0 R      02
A787                      000009C7 R      02
A8600                     0000098A R      02
ACNT_NAME                 00000000 R      02
ACTRTN                    00000C40 R      05
ALLSPOOL                  00000783 R      02
ANAUTILUS                 00000997 R      02
ANSWER                    00000857 R      03
ARG_COUNT                 00000436 R      03
ASCORPIO                  0000098F R      02
ASTLM                     00000854 R      02
AUV1                      000009A0 R      02
AUV2                      000009AB R      02
BIOLM                     0000085A R      02
BUFFER                    0000004D R      03
BUFFER_PTR                00000045 R      03
BUGCHK                    0000078C R      02
BYPASS                    00000793 R      02
CCASTHAND                 00000CE7 R      05
CHF$L_SIGARGLST         = 00000004
CHF$L_SIG_ARG1          = 00000008
CHF$L_SIG_ARGS          = 00000000
CHF$L_SIG_NAME          = 00000004
CLI$K_CLISERV           = 00000005
CLI$K_GETSYM            = 0000000A
CLI$K_LOCAL_SYM         = 00000001
CLI$Q_NAMDESC           = 00000004
CLI$Q_VALDESC           = 0000000C
CLI_REQ_DESC              000009FB R      03
CMEXEC                    0000079A R      02
CMKRNL                    000007A1 R      02
CNTRLCMSG                 000000B8 R      02
COMMAND_DVI_FAILED        000001A6 R      02
COMMAND_ITMLST            0000004B R      02
COMMA_BLANK               00000ACC R      02
COMMON                    00000C9A R      05
CPULIM                    00000860 R      02
CPU_NAME_TABLE            000008E5 R      02
CPU_SCALE                 00000993 R      03
CPU_SCALE_DES             000002DE R      03
CPU_SCALE_TABLE           00000941 R      02
CPU_TYPE_TABLE            000008AA R      02
CR                      = 0000000D
CTRSTR                    000002A1 R      02
DCT_LENGTH              = 00000009
DCS_TERM                  ********     X  05
DETACH                    000007A8 R      02
DEVBUF                    00000401 R      03
DIAGNOSE                  000007AF R      02
DIOLM                     0000086D R      02
DISK                      0000033F R      02
DISK_BUFFER               000002FA R      03
DUMP                      0000002C R      02
DUMPM                   = 00000010
DUMPV                   = 00000004
DUMP_MSG1                 0000048E R      02
DUMP_MSG2                 00000554 R      02
DUMP_MSG_PTR              0000047E R      02
DVI$_DEVCLASS             ********    X  02
DVI$_DEVNAM               ********    X  02
ENDSTR                    00000250 R      02
ENQLM                     00000867 R      02
ERROR_COUNT               00000039 R      03
ERROR_EXIT                00000D1E R      05
EXE$GL_CONFREGL           ********    X  05
EXE$GL_MP                 ********    X  05
EXE$GL_RPB                ********    X  05
EXIT_DESC                 00000426 R      03
EXIT_HANDLER              00000D7A R      05
EXPECTED                  0000064B R      02
EXQUOTA                   000007B8 R      02
FAB$B_BID               = 00000000
FAB$B_FNS               = 00000034
FAB$C_BID               = 00000003
FAB$C_BLN               = 00000050
FAB$C_SEQ               = 00000000
FAB$C_VAR               = 00000002
FAB$L_ALQ               = 00000010
FAB$L_FNA               = 0000002C
FAB$L_FOP               = 00000004
FAB$L_STS               = 00000008
FAB$L_STV               = 0000000C
FAB$V_CHAN_MODE         = 00000002
FAB$V_CR                = 00000001
FAB$V_FILE_MODE         = 00000004
FAB$V_LNM_MODE          = 00000000
FAB$V_PUT               = 00000000
FAB$W_GBC               = 00000048
FAO_BOF                   0000003D R      03
FILE                      000002C3 R      02
FILLM                     00000873 R      02
FINAL_MESSAGE             000009B8 R      05
FINI                      00000D57 R      05
FLAGS                     0000043A R      03
FOR$CNV_OUT_F             ********  X     05
GETSYI_ITMLST             0000088E R      02
GET_MEM_INFO              00000AC7 R      05
GROUP                     000007C0 R      02
GRPNAM                    000007C6 R      02
INVALID_LOADCNT_MSG       00000159 R      02
INVALID_PASS_MSG          00000133 R      02
```

```
INVALID_PHASE_MSG               00000100 R      02
INVALID_REPORT_MSG              0000017F R      02
IO$M_CTRLCAST                   ******** X      05
IO$_SETMODE                     ******** X      05
JPI$_ASTLM                    = 00000409
JPI$_BIOLM                    = 00000310
JPI$_BYTLM                    = 0000031A
JPI$_CPULIM                   = 0000040D
JPI$_CURPRIV                  = 00000400
JPI$_DIOLM                    = 00000313
JPI$_ENQLM                    = 00000320
JPI$_FILLM                    = 0000040F
JPI$_PGFLQUOTA                = 0000040E
JPI$_PRCLM                    = 00000408
JPI$_TQLM                     = 00000410
JPI$_USERNAME                 = 00000202
JPI$_WSQUOTA                  = 00000402
JPI_ASTLM                       000009B3 R      03
JPI_BIOLM                       000009B7 R      03
JPI_BYTLM                       000009B8 R      03
JPI_CPULIM                      000009BF R      03
JPI_DIOLM                       000009C7 R      03
JPI_ENQLM                       000009C3 R      03
JPI_FILLM                       000009CB R      03
JPI_PGFLQUOTA                   000009CF R      03
JPI_PRCLM                       000009D3 R      03
JPI_TQLM                        000009D7 R      03
KEY_ALL_DESC                    00000B41 R      02
KEY_CLUSTER_DESC                00000B98 R      02
KEY_DECNET_DESC                 00000B90 R      02
KEY_DEVICE_DESC                 00000B80 R      02
KEY_LOAD_DESC                   00000B88 R      02
KEY_SUBSET_DESC                 00000B49 R      02
LCBIT                         = 00000020
LF                            = 0000000A
LIB$$A_HERE                   = 00000B80 R      02
LIB$$A_STRLOC                 = 00000B88 R      02
LIB$$K_NPAIRS                 = 00000004
LIB$GET_COMMAND                 ******** X      05
LIB$LOOKUP_KEY                  ******** X      05
LIB$SIGNAL                      ******** X      05
LOAD                            000006E3 R      05
LOADS_DESC                      000002D6 R      03
LOAD_COUNT                      0000099B R      03
LOAD_MSG                        0000018D R      03
LOAD_PROMPT                     0000039E R      02
LOGINOUT                        00000605 R      02
LOGNAM_SIZE                   = 000000FF
LOG_FAB                         00000A10 R      03
LOG_IO                          000007CD R      02
LOG_RAB                         00000A60 R      03
LONG_MSG                        0000044D R      02
M                             = 0000004D
MAXSYM_SZ                     = 000000FF
MEM_FREE                        000009E7 R      03
MEM_MODIFY                      000009EB R      03
MEM_SIZE                        000009E3 R      03
MMG$GL_PHYPGCNT                 ********   X    05
MODE                            00000020 R      02
MOUNT                           000007D4 R      02
MSG_BLOCK                       00000405 R      03
MSG_DESC                        000009AB R      03
MTH$JMINO                       ********   X    05
NAME_TBL                        00000783 R      02
NAM_PTRS                        000006E7 R      02
NDT$_MPM0                     = 00000040
NDT$_MPM1                     = 00000041
NDT$_MPM2                     = 00000042
NDT$_MPM3                     = 00000043
NETMBX                          000007DA R      02
NEW_LINE                        00000ACF R      02
NOACNT                          000007E1 R      02
NO_RMS_AST_TABLE                000000A4 R      02
NRAT_LENGTH                   = 00000014
OFFSET                          00000624 R      02
OPER                            000007E8 R      02
OTS$CVT_L_TI                    ********   X    05
OTS$CVT_TI_L                    ********   X    05
OTS$_INPCONERR                  ********   X    05
OUTLEN                          00000988 R      03
P1_BUF                          0000045B R      03
P1_DESC                         0000043B R      03
P1_LEN                        = 00000002
P1_NAM                          00000A8E R      02
P2_BUF                          0000055A R      03
P2_DESC                         00000443 R      03
P2_LEN                        = 00000002
P2_NAM                          00000A90 R      02
P3_BUF                          00000659 R      03
P3_DESC                         0000044B R      03
P3_LEN                        = 00000002
P3_NAM                          00000A92 R      02
P4_BUF                          00000758 R      03
P4_DESC                         00000453 R      03
P4_LEN                        = 00000002
P4_NAM                          00000A94 R      02
PAGE_BUF                        0000040D R      03
PAGE_COUNT                      00000409 R      03
PAGE_SIZE                       000009F3 R      03
PARAM_BUF                       000001AA R      03
PARAM_MSG                       000001A2 R      03
PASS                            00000626 R      05
PASS_COUNT                      00000997 R      03
PASS_MSG                        00000179 R      03
PASS_NAME                       00000074 R      02
PASS_PROMPT                     00000366 R      02
PC1...                        = 0000064B R      02
PC2...                        = 000006E7 R      02
PC3...                        = 00000783 R      02
PC5...                        = 0000088E R      02
PER_WS_INUSE                  = CCCD3F4C
PFNMAP                          000007ED R      02
PGFLQUOTA                       00000879 R      02
PHASE                           000003A6 R      05
```

```
PHASES                          0000043E R      02
PHASE_PROMPT                    00000A96 R      02
PHASE_TABLE                     00000B5C R      02
PHY_IO                          000007F4 R      02
PP_PAGE_USAGE                 = 000003E8
PR$S_SID_TYPE                 = 00000008
PR$V_SID_TYPE                 = 00000018
PR$_SID_TYP730                = 00000003
PR$_SID_TYP750                = 00000002
PR$_SID_TYP780                = 00000001
PR$_SID_TYP790                = 00000004
PR$_SID_TYPUV1                = 00000007
PR$_SID_TYPUV2                = 00000008
PRCLM                           00000883 R      02
PRIVS                           000009DB R      03
PRIV_CNT                      = 0000001E
PRIV_PRNTV                    = 00000003
PRMCEB                          000007FB R      02
PRMGBL                          00000802 R      02
PRMMBX                          00000809 R      02
PROMPTM                       = 00000002
PROMPTV                       = 00000001
PRV_STR                         000002B3 R      02
PSWAPM                          00000810 R      02
QUAD_STATUS                     0000041A R      03
QUOT_CNT                      = 00000009
QUO_STR                         000002BD R      02
RAB$B_RAC                     = 0000001E
RAB$C_BID                     = 00000001
RAB$C_BLN                     = 00000044
RAB$C_SEQ                     = 00000000
RAB$L_CTX                     = 00000018
RAB$L_FAB                     = 0000003C
RAB$L_RBF                     = 00000028
RAB$L_ROP                     = 00000004
RAB$L_STS                     = 00000008
RAB$L_STV                     = 0000000C
RAB$W_RSZ                     = 00000022
RECORD                          000002CF R      02
REPORT_NAME                     00000083 R      02
REPORT_PROMPT                   000003D7 R      02
REPORT_Q                        000008EB R      05
RMS$_BLN                        ********   X    02
RMS$_BUSY                       ********   X    02
RMS$_CDA                        ********   X    02
RMS$_FAB                        ********   X    02
RMS$_FACILITY                 = 00000001
RMS$_RAB                        ********   X    02
RMS_ERROR                       00000C64 R      05
RMS_ERR_STRING                  000002DD R      02
RPB$C_MEMDSCSIZ               = 00000008
RPB$L_BOOTR5                  = 00000030
RPB$L_MEMDSC                  = 000000BC
RPB$S_PAGCNT                  = 00000018
RPB$S_TR                      = 00000008
RPB$V_MPM                     = 0000000B
RPB$V_PAGCNT                  = 00000000
RPB$V_TR                      = 00000018
RPB$V_USEMPM                  = 0000000C
SCH$GL_FREECNT                  ********   X    05
SCH$GL_MFYCNT                   ********   X    05
SCH$GW_PROCCNT                  ********   X    05
SCH$GW_PROCLIM                  ********   X    05
SELECT_PHASE                    00000B2D R      02
SETPRI                          00000817 R      02
SETPRV                          0000081E R      02
SHMEM                           00000825 R      02
SHORT_MSG                       00000465 R      02
SHR$_ABENDD                   = 000010E0
SHR$_BADKEY                   = 00001108
SHR$_BEGIND                   = 00001038
SHR$_ENDEDD                   = 00001080
SHR$_TEXT                     = 00001130
SID                             000009F7 R      03
SPACE                         = 00000020
SS$_BADPARAM                    ********   X    05
SS$_CONTROLC                    ********   X    05
SS$_NORMAL                      ********   X    05
SS$_SSFAIL                      ********   X    05
SS$_WASSET                      ********   X    05
SSERROR                         00000B5B R      05
START_MESSAGE                   0000040F R      02
STATUS                          00000422 R      03
STR$UPCASE                      ********     X  05
STRSTR                          00000238 R      02
STS$K_ERROR                   = 00000002
STS$K_SUCCESS                 = 00000001
STS$K_WARNING                 = 00000000
STS$M_INHIB_MSG               = 10000000
STS$S_FAC_NO                  = 0000000C
STS$S_SEVERITY                = 00000003
STS$V_FAC_NO                  = 00000010
STS$V_SEVERITY                = 00000000
SWAP_SIZE                       000009EF R      03
SYI$_PAGEFILE_FREE            = 000010F4
SYI$_SID                      = 00001001
SYMBOL_CNT                    = 00000004
SYM_NAM_TABLE                   00000A6E R      02
SYM_P1                          00000A6E R      02
SYM_P2                          00000A76 R      02
SYM_P3                          00000A7E R      02
SYM_P4                          00000A86 R      02
SYM_VAL_TABLE                   0000043B R      03
SYNTAX_ERROR                    00000CD3 R      05
SYNTAX_ERROR_MSG                000000D9 R      02
SYS$ASSIGN                      ********  GX    05
SYS$CLI                         ********     X  05
SYS$CLOSE                       ********  GX    05
SYS$CMEXEC                      ********  GX    05
SYS$COMMAND                     00000038 R      02
SYS$CONNECT                     ********  GX    05
SYS$CREATE                      ********  GX    05
SYS$CRELOG                      ********  GX    05
SYS$DCLEXH                      ********  GX    05
```

```
SYS$DELLOG                      ********  GX    05
SYS$EXIT                        ********  GX    05
SYS$FAO                         ********   X    05
SYS$FAOL                        ********  GX    05
SYS$GETDVI                      ********  GX    05
SYS$GETJPI                      ********  GX    05
SYS$GETMSG                      ********  GX    05
SYS$GETSYI                      ********  GX    05
SYS$GQ_VERSION                  ********   X    05
SYS$PUT                         ********  GX    05
SYS$PUTMSG                      ********  GX    05
SYS$QIOW                        ********  GX    05
SYS$SETAST                      ********  GX    05
SYS$SETPRN                      ********  GX    05
SYS$SETSFM                      ********  GX    05
SYS$TRNLOG                      ********  GX    05
SYSDISK                         00000091  R     02
SYSGBL                          0000082B  R     02
SYSNAM                          00000832  R     02
SYSPRV                          00000839  R     02
SYSTEM                          000002FE  R     02
TAB                         =   00000009
TERMINALM                   =   00000004
TERMINALV                   =   00000002
TEST_NAME                       0000000F  R     02
TEXT_BUFFER                 =   0000012C
TMPMBX                          00000840  R     02
TQLM                            00000889  R     02
TTCHAN                          00000037  R     03
UETINITOO                       00000000  RG    05
UETP                        =   00740000
UETP$$A_THERE               =   00000BA0  R     02
UETP$_ABENDD                =   007410E0
UETP$_ABORTC                =   0074832B
UETP$_BADKEY                =   00741108
UETP$_BEGIND                =   00741038
UETP$_ENDEDD                =   00741080
UETP$_ERBOXPROC             =   00748020
UETP$_FACILITY              =   00000074
UETP$_TEXT                  =   00741130
UETPPHASE                       00000BBB  R     02
UNKNOWN_CPU                     0000096D  R     02
USERS                           00000067  R     02
USER_LIST                       000009CE  R     02
VECTOR                          0000099F  R     03
VERSION                         0000002D  R     03
VOLPRO                          00000847  R     02
WELCOME                         00000000  R     03
WELCOML                     =   0000002F
WHICH_PHASE1                    00000AD3  R     02
WHICH_PHASE2                    00000B1E  R     02
WORLD                           0000084E  R     02
WRONG_ACCOUNT                   000001E3  R     02
WS_INUSE                        000002F6  R     03
WS_INUSE_DES                    000002EA  R     03
WS_SIZE                         000009AF  R     03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	00000BCC (3020.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00000AA4 (2724.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
$RMSNAM	00000008 (8.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
UETINITOO	00000D9C (3484.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:00.44
Command processing	108	00:00:00.68	00:00:02.67
Pass 1	574	00:00:24.52	00:00:50.29
Symbol table sort	0	00:00:02.47	00:00:04.45
Pass 2	386	00:00:06.94	00:00:13.70
Symbol table output	39	00:00:00.35	00:00:00.93
Psect synopsis output	0	00:00:00.03	00:00:00.03
Cross-reference outout	0	00:00:00.00	00:00:00.00
Assembler run totals	1138	00:00:35.08	00:01:12.52

The working set limit was 2000 pages.
144004 bytes (282 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1681 non-local and 74 local symbols.
2056 source lines were read in Pass 1, producing 50 object records in Pass 2.
61 pages of virtual memory were used to define 54 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]UETP.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	2
_255DUA28:[SYSLIB]STARLET.MLB;2	47
TOTALS (all libraries)	50

1898 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:UETINITOO/OBJ=OBJ$:UETINITOO MSRC$:UETINITOO/UPDATE=(ENH$:UETINITOO)+EXECML$/LIB+SHRLIB$:UETP/LIB
```